- Networks become unreachable.
- Broadcast traffic increases.

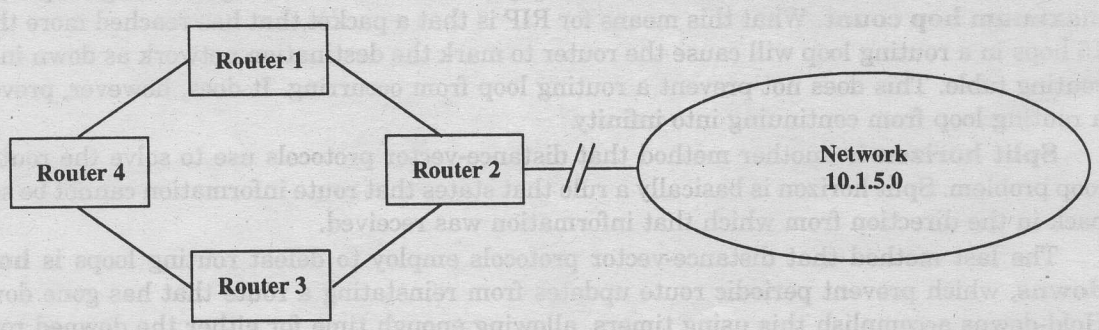
### Increased Convergence Time

Convergence takes so long because the routers now have to wait until another router volunteers a pathway to the down network. But before another router can volunteer a pathway, the routers must wait a set time limit to prevent routing loops. Convergence time increases as the network grows, because more routers need to be updated with route information. The more time it takes for the internetwork to converge, the slower the internetwork is perceived by end users. This is not a problem when there are only a few routers that need to be updated.

### Routing Loops

Routing loops occur because routers on an internetwork are not updated at close to the same time. This causes routers to send outdated route information as though the information were new. Figure 12.9 shows a network before and after a link failure was detected. When Router 2 detects a link failure to network 10.1.5.0, it updates its routing table and sends its entire route table to the other routers on the internetwork. Router 3 receives the update from Router 2 and updates its routing table. At this point, any packets received on Router 3 destined for 10.1.5.0 will not be sent to Router 2.

Router 1 still has not received an update for network 10.1.5.0. Since Router 1 thinks 10.1.5.0 can still be reached via Router 4, this incorrect route information is sent in the route update to the other routers on the internetwork. Router 4 receives the update from Router 1 and updates its routing table. Before Router 1 receives an update that network 10.1.5.0 is down, it has already sent out updates letting other routers know that it has a live route to 10.1.5.0. The other routers send packets destined for 10.1.5.0 to Router 1, while Router 1 sends the same packets to Router 4. This creates a routing loop. This is not a problem on small internetworks with a few routers, because the routers are updated at close to the same time.




| | | | |
|---|---|---|---|
| Before Link Down Detected<br>Router 1 Route Table<br>10.1.5.0 via Router 4 | Router 2 Route Table<br>10.1.5.0 via local interface | Router 3 Route Table<br>10.1.5.0 via Router 2 | Router 4 Route Table<br>10.1.5.0 via Router 3 |
| After Link Down Detected<br>Router 1 Route Table<br>10.1.5.0 via Router 4 | Router 2 Route Table<br>10.1.5.0 via Router 3 | Router 3 Route Table<br>10.1.5.0 via Router 4 | Router 4 Route Table<br>10.1.5.0 via Router 1 |

**Fig. 12.9** Routing loop

### Unreachable Networks

If distance-vector routing protocols are in use, internetwork growth can cause some networks to be unreachable. This is due to the hop-count limitation of distance-vector protocols. For example, RIP has a maximum hop count of 15. Any network more than 15 hops away is considered unreachable by RIP. Network growth should be monitored closely and a periodic re-evaluation of routing protocols should be done.

### Increased Broadcasts

Distance-vector protocols are known for being "chatty". In other words, they put a constant flow of traffic on the internetwork. This traffic comes in the form of periodic broadcasts containing the entire route tables sent from routers running distance-vector protocols. The larger a network gets, the more routers there are to load the internetwork with broadcasts. Eventually, this process can cause varying degrees of slowness on the internetwork as network devices and end stations compete for bandwidth.

Organizations with large internetworks steer clear of protocols that operate using excessive broadcasts. The reason for this avoidance is that too many broadcasts can degrade network performance. When designing large internetworks, use routed protocols such as IP and routing protocols such as EIGRP or OSPF. This solution will allow your design to be more scalable and lacking in excessive broadcasts.

### Possible solutions to the problems of distance vector protocols

Even though distance-vector protocols are plagued with routing issues as an internetwork grows, these protocols do have methods for functioning in these larger environments. These methods include the usage of timers and rules for preventing routing loops. Of course, if an internetwork grows too large and too complex, link-state routing protocols should be considered.

One method that distance-vector routing protocols use to manage routing loops is a **maximum hop count**. What this means for RIP is that a packet that has reached more than 15 hops in a routing loop will cause the router to mark the destination network as down in its routing table. This does not prevent a routing loop from occurring. It does, however, prevent a routing loop from continuing into infinity.

**Split horizon** is another method that distance-vector protocols use to solve the routing loop problem. Split horizon is basically a rule that states that route information cannot be sent back in the direction from which that information was received.

The last method that distance-vector protocols employ to defeat routing loops is **hold-downs**, which prevent periodic route updates from reinstating a route that has gone down. Hold-downs accomplish this using timers, allowing enough time for either the downed route to come back online or the network to stabilize before switching to the next best route. Hold-down timers are reset by triggered updates under the following circumstances:

- The hold-down timer expires.
- The router receives a processing task that is proportional to the number of links in the internetwork.
- The router receives another update indicating a change in the network's status.

**Advantages of Distance Vector-Based Routing Protocols**

- **Simpler :** Distance vector-based routing protocols are simple router advertisement processes that are easy to understand.
- **Easy to configure :** In its simplest incarnation, configuring a distance vector-based routing protocol is as easy as enabling it on the router interfaces.

**Disadvantages of Distance Vector-Based Routing Protocols**

- **Large routing tables :** Multiple routes to a given network ID can be reflected as multiple entries in the routing table. In a large internetwork with multiple paths, the routing table can have hundreds or thousands of entries.
- **High network traffic overhead :** Route advertising is done periodically even after the internetwork has converged.
- **Does not scale :** Between the size of the routing table and the high overhead, distance vector-based routing protocols do not scale well to large and very large internetworks.
- **High convergence time :** Due to the unsynchronized and unacknowledged way that distance vector information is exchanged; convergence of the internetwork can take several minutes. While converging, routing loops and black holes can occur.

### 12.4.7.2 Link-state routing

In distance-vector routing, the information about the topology of the network is distributed among the routers, which achieves efficiency but also creates problems. An alternate approach, used in link-state routing, is to give each router complete information about the graph of the network. Then, each router independently computes the optimal path to every destination.

The information available to a distance vector router has been compared to the information available from a road sign. Link state routing protocols are like a road map. A link state router cannot be fooled as easily into making bad routing decisions, because it has a complete picture of the network. The reason is that unlike the routing-by-rumor approach of distance vector, link state routers have first hand information from all their peer routers. Each router originates information about itself, its directly connected links, and the state of those links (hence the name). This information is passed around from router to router, each router making a copy of it, but never changing it. The ultimate objective is that every router has identical information about the internetwork and each router will independently calculate its own best paths.

Link State Routing protocols provide greater flexibility and sophistication than their Distance Vector counterparts. They reduce overall broadcast traffic and make better decisions about routing by taking characteristics such as bandwidth, delay, reliability, and load into consideration instead of basing their decisions solely on distance or hop count.

Link-State Routing reduces network traffic. Routers newly attached to network request information from nearby routers. Once information has been exchanged, routers only broadcast when something has changed. These messages contain information about the state

of each link that the router has with other routers. They keep each other updated and therefore complete updates are seldom needed.

Link state protocols, sometimes called shortest path first or distributed database protocols, are built around a well-known algorithm from graph theory, E. W. Dijkstra's shortest path algorithm. Examples of link state routing protocols are:

- Open Shortest Path First (OSPF)
- The ISO's Intermediate System to Intermediate System (IS-IS)
- DEC's DNA Phase V
- Novell's NetWare Link Services Protocol (NLSP)

Although link state protocols are rightly considered more complex than distance vector protocols, the basic functionality is not complex at all:

1. Each router establishes a relationship—an adjacency—with each of its neighbors.
2. Each router sends *Link State Advertisements* (LSAs).
3. Each router stores a copy of all the LSAs it has seen in a database. If all works well, the databases in all routers should be identical.
4. The completed *topological database*, also called the *link state database*, describes a graph of the internetwork. Using the Dijkstra algorithm, each router calculates the shortest path to each network and enters this information into the route table.

**General Link State Operation**

Link State Routing protocols reduce broadcast traffic because they do not send out periodic broadcasts or send out their entire tables with each broadcast. Link state routing protocols exchange a complete copy of their route tables upon initialization. Thereafter route updates are multicast only when a change has occurred (triggered by a change in the topology), including only the change in the update not the entire table. Changes are flooded immediately and computed in parallel. If no changes occur, they do not generate an update.

Triggered updates improve convergence time by requiring routers to send an update message immediately upon learning of a route change. These updates are triggered by some event, such as a new link becoming available or an existing link failing.

All Link State protocols must build and maintain three separate tables (also referred to as databases):

- The neighbor table (also known as Adjacency database)
- Topology Map (also known as Link State database)
- Route Table (also known as Forwarding database)

Before traffic can be forwarded certain things must happen.

1. All routers attached to the same network must identify themselves and their neighbors, establishing a relationship with one another (referred to as adjacency). Adjacent routers form this relationship through the exchange of hello messages. From these hellos each local router builds its first table (the neighbor). Fig. 12.10(a)

shows the neighbor hello exchange and figure 12.10(b) shows a routers neighbor table. If this first stage fails, routing will never occur.

**Note :** Within hello messages routers announce themselves; identify the links they are directly attached to and the state of these links (up or down), among other things.).

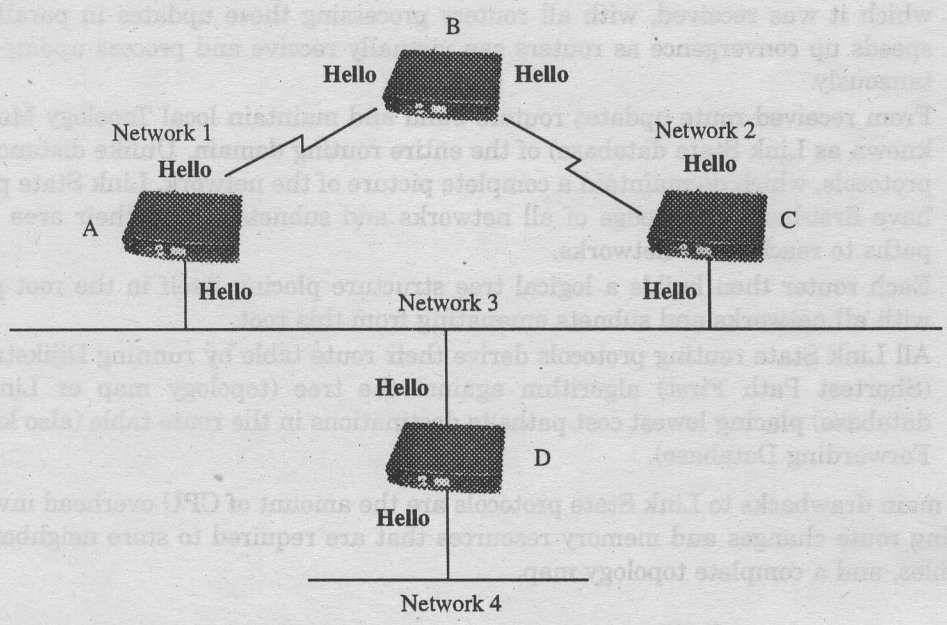



**Fig. 12.10 (a)** Link state routers exchange hello messages

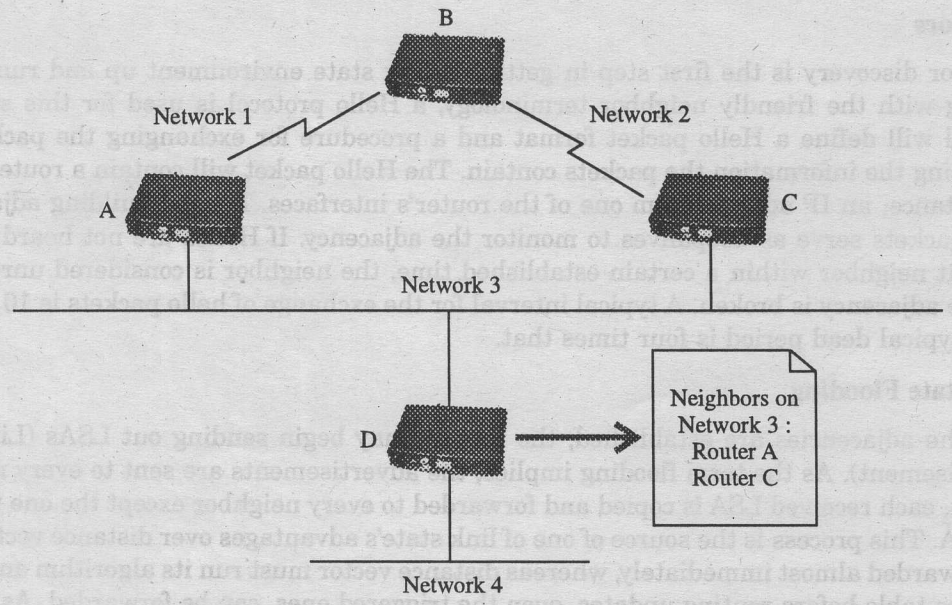



**Fig. 12.10 (b)** Router D has discovered two neighbors on network 3

2. Once the adjacency relationship is intact, route information may be exchanged by flooding updates throughout the routing domain. Unlike Distance Vector routing protocols, which must receive, process and wait for their periodic timer to expire before sending route updates on, any change in the network triggers Link State protocols to immediately flood route updates to all segments except the one from which it was received, with all routers processing these updates in parallel. This speeds up convergence as routers can virtually receive and process updates simultaneously.
3. From received route updates routers build and maintain local Topology Maps (also known as Link State database) of the entire routing domain. Unlike distance vector protocols, which do maintain a complete picture of the network, Link State protocols have first-hand knowledge of all networks and subnets within their area and the paths to reach these networks.
4. Each router then builds a logical tree structure placing itself in the root position, with all networks and subnets emanating from this root.
5. All Link State routing protocols derive their route table by running Dijikstra's SPF (Shortest Path First) algorithm against the tree (topology map or Link State database) placing lowest cost paths to destinations in the route table (also known as Forwarding Database).

The main drawbacks to Link State protocols are the amount of CPU overhead involved in calculating route changes and memory resources that are required to store neighbor tables, route tables, and a complete topology map.

## Common Characteristics of Link State Routing

### Neighbors

Neighbor discovery is the first step in getting a link state environment up and running. In keeping with the friendly neighbor terminology, a Hello protocol is used for this step. The protocol will define a Hello packet format and a procedure for exchanging the packets and processing the information the packets contain. The Hello packet will contain a router ID and the instance, an IP address from one of the router's interfaces. Beyond building adjacencies, Hello packets serve as keepalives to monitor the adjacency. If Hellos are not heard from an adjacent neighbor within a certain established time, the neighbor is considered unreachable and the adjacency is broken. A typical interval for the exchange of hello packets is 10 seconds, and a typical dead period is four times that.

### Link State Flooding

After the adjacencies are established, the routers may begin sending out LSAs (Link State Advertisement). As the term flooding implies, the advertisements are sent to every neighbor. In turn, each received LSA is copied and forwarded to every neighbor except the one that sent the LSA. This process is the source of one of link state's advantages over distance vector. LSAs are forwarded almost immediately, whereas distance vector must run its algorithm and update its route table before routing updates, even the triggered ones, can be forwarded. As a result,

link state protocols converge much faster than distance vector protocols converge when the topology changes.

### The Link State Database

In addition to flooding LSAs and discovering neighbors, a third major task of the link state routing protocol is establishing the link state database. The link state or topological database stores the LSAs as a series of records. The important information for the shortest path determination process is the advertising router's ID, its attached networks and neighboring routers, and the cost associated with those networks or neighbors. So LSAs may include two types of generic information:

- *Router link information* advertises a router's adjacent neighbors with a triple of (Router ID, Neighbor ID, Cost), where cost is the cost of the link to the neighbor.
- *Stub network information* advertises a router's directly connected stub networks (networks with no neighbors) with a triple of (Router ID, Network ID, Cost).

### Example of Link State Algorithm

In this algorithm, a router, based on information that has been collected from other routers, builds a graph of the network. This graph shows the location of routers in the network and their links to each other. Every link is labeled with a number called the **weight** or **cost**. This number is a function of delay time, average traffic, and sometimes simply the number of hops between nodes. For example, if there are two links between a node and a destination, the router chooses the link with the lowest weight.

The **Dijikstra algorithm** goes through these steps:

1. The router builds a graph of the network and identifies source and destination nodes, as V1 and V2 for example. Then it builds a matrix, called the "adjacency matrix". In this matrix, a coordinate indicates weight. For example, [i, j] is the weight of a link between $V_i$ and $V_j$. If there is no direct link between $V_i$ and $V_j$, this weight is identified as "infinity".
2. The router builds a status record set for every node on the network. The record contains three fields:
   - Predecessor field—The first field shows the previous node.
   - Length field—The second field shows the sum of the weights from the source to that node.
   - Label field —The last field shows the status of node. Each node can have one status mode: "permanent" or "tentative".
3. The router initializes the parameters of the status record set (for all nodes) and sets their length to "infinity" and their label to "tentative".
4. The router sets a T-node. For example, if V1 is to be the source T-node, the router changes V1's label to "permanent". When a label changes to "permanent", it never changes again. A T-node is an agent and nothing more.
5. The router updates the status record set for all tentative nodes that are directly linked to the source T-node.

6. The router looks at all of the tentative nodes and chooses the one whose weight to $V_1$ is lowest. That node is then the destination T-node.
7. If this node is not V2 (the intended destination), the router goes back to step 5.
8. If this node is V2, the router extracts its previous node from the status record set and does this until it arrives at V1. This list of nodes shows the best route from V1 to V2.

These steps are shown below as a flowchart.

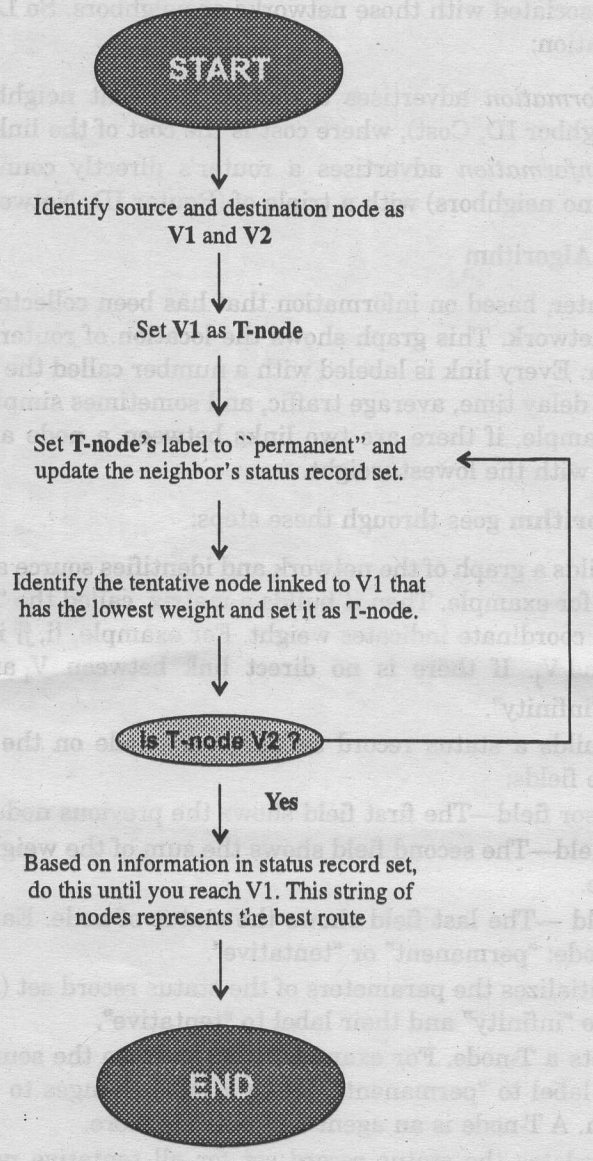



**Fig. 12.11** Dijikstra algorithm

***Example*** : Suppose we have a graph of network build according to link state algorithm as shown in figure 12.12 (a) and we are to find the best route between A and E. We can see that there are six possible routes between A and E (ABE, ACE, ABDE, ACDE, ABDCE, ACDBE). We have to use the above algorithm to find the best route.

1. The source node (A) has been chosen as T-node, and so its label is permanent.

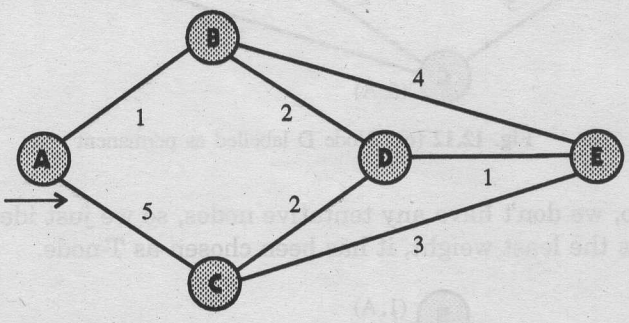



**Fig. 12.12 (a)** A network graph—Node A labelled as permanent

2. In this step, we see that the status record set of tentative nodes directly linked to T-node (B, C) has been changed. Also, since B has less weight, it has been chosen as T-node and its label has changed to permanent (see below).

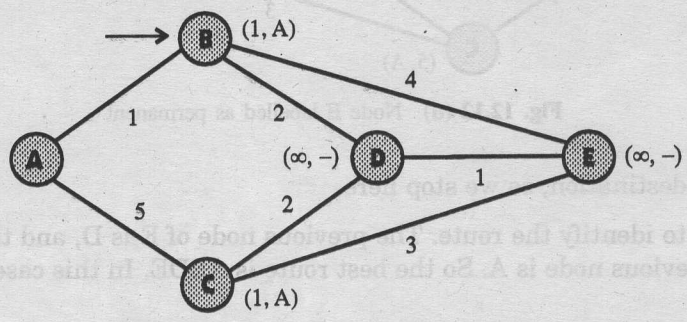



**Fig. 12.12 (b)** Node B labelled as permanent

3. In this step, like in step 2, the status record set of tentative nodes that have a direct link to T-node (D, E), has been changed. Also, since D has less weight, it has been chosen as T-node and its label has changed to permanent (see fig. 12.12 (c)).

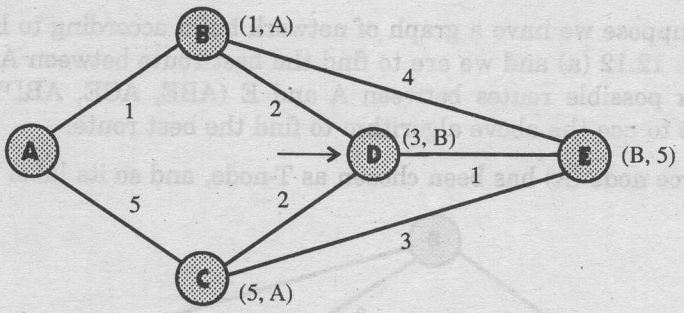



**Fig. 12.12 (c)** Node D labelled as permanent

4. In this step, we don't have any tentative nodes, so we just identify the next T-node. Since E has the least weight, it has been chosen as T-node.

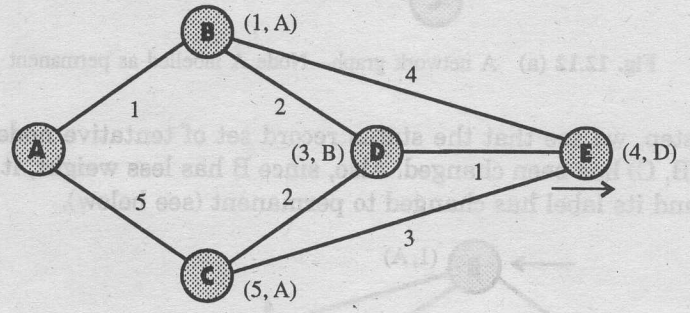



**Fig. 12.12 (d)** Node E labelled as permanent

5. E is the destination, so we stop here.

Now we have to identify the route. The previous node of E is D, and the previous node of D is B, and B's previous node is A. So the best route is ABDE. In this case, the total weigh is 4(1 + 2 + 1).

Although this algorithm works well, it's so complicated that it may take a long time for routers to process it, and the efficiency of the network fails. Also, if a router gives the wrong information to other routers, all routing decisions will be ineffective.

**Advantages of Link State-Based Routing Protocols**

- **Smaller routing tables :** Only a single optimal route for each network ID is stored in the routing table.
- **Low network overhead :** Link state based routers do not exchange any routing information when the internetwork has converged.
- **Ability to scale :** Between the smaller routing tables and low overhead, link state based routing protocols scale well to large and very large internetworks.
- **Lower convergence time :** Link state based routing protocols have a much lower convergence time and the internetwork is converged without routing loops.

**Disadvantages of Link State-Based Routing Protocols**

- **Complex :** Link state based routing protocols are much more complex and difficult to understand than distance vector based routing protocols.
- **More difficult to configure :** A link state based routing protocol implementation requires additional planning and configuration.
- **Resource intensive :** For very large internetworks, the database of link state advertisements and the calculation of routing table entries can be memory and processor intensive.

**Potential Problems**

- **Processing and memory required**—More memory is required due to the large amount of network information that needs to be managed. Recall that each router has a complete picture of the network.
- **Bandwidth consumed for initial link state "flood"**—A lot of bandwidth is consumed during initial link-state startup. This issue is called flooding. This is caused from all routers trying to converge the network and understand the network topology.
- **Unsynchronized updates and inconsistent path decisions**—This often happens when links become unavailable and the LSP has to be reconstructed. During the calculation other links go down as other come up. This creates problems.
- **Large network synchronization**—This adds to the problems due to the complete network topology that each router contains. Larger networks will run Hybrid protocols.

**Implemented Solutions**

- Reduce the need for resources.
- Coordinate link-state updates.

### 12.4.8 Comparing Distance-Vector Protocols to Link-State Routing Protocols

Both types have their place in internetworks. Distance-vector protocols were created when internetworks spanning hundreds of routers were uncommon. Link-state protocols, on the other hand, were specifically designed for these larger internetworks.

- **Distance-Vector**
  - Tell neighbors about distances to all destinations.
  - Node's computation depends on neighbors.
- **Link-State**
  - Tell all routers distance to each neighbor.
  - Each router computes its best paths.
- **Both are distributed algorithms**

The terms 'distance vector' and 'link state' are used to group routing protocols into two

broad groups based on whether the protocol makes shortest path decisions based on a hop count metric or cost-based metrics. In distance-vector each node talks only to direct neighbors telling them all it knows. In link-state each node talks to all nodes telling them about direct neighbors.

**Distance Vector**

Distance vector protocols count the number of devices data must flow through to reach a destination. Each device is referred to as a 'hop', so the total number of hops between source and destination is the 'hop count'. Routes with lower hop counts are preferred.

Distance Vector algorithms require very little overhead in terms of processor power and memory, compared to Link State. Distance vector algorithms calculate only the hop count to destinations and choose best routes according to the hop count metric.

**Link State**

Link State protocols track the status and connection type (and therefore speed) of each link, and produce a calculated metric based on these and other factors, including some set by the network administrator. Distance Vector protocols differ from link state protocols because they see all hops as equal and only the number of hops matters. Link State protocols will take a path which has more hops, but that uses a faster medium over a path using a slower medium with fewer hops.

Because of their awareness of media types and other factors, link state protocols require more processing power (more circuit logic) and memory. Distance vector algorithms being simpler require simpler hardware.

**When should distance-vector be used, and when should link-state routing be used?**

There are arguments for and against each approach. The choice of strategy should be based on the properties of the given network, the main parameter being how frequently nodes are expected to go down or come up.

**Stability**

1. It is generally thought that link-state algorithms are more stable, because each router "knows" the entire network.
2. On the other hand, if the network is highly dynamic, it is possible for link-state algorithms to create routing loops while new topology information is being disseminated.

**Running Time**

1. Link-state algorithms converge more quickly than distance-vector algorithms.
2. In practice, the time for convergence depends on the network graph, the load on the routing protocol, and the exact sequence of link failure and recovery.

**Overhead**

1. Much of the overhead in link-state routing comes from the subroutine that prevents

corruption of the LSP database in case of node and/or link failure. The distance-vector algorithm does not have this overhead.

**Memory**

1. Typically, link-state implementations require more memory than distance-vector implementations.

**The table below compares link-state protocols to distance-vector protocols:**

| Distance-Vector Protocol | Link-State Protocol |
|---|---|
| Must trust route tables of other routers; not always accurate. | Compiles an accurate topology map. |
| Calculates best path using hops. | Uses bandwidth analysis and other metrics for best-path calculation. |
| Updates only occur at preset intervals. | Updates occur whenever there is a link change. |
| Requires less memory. | Requires more memory. |
| Requires less processor utilization. | Requires more processor utilization. |
| Requires more bandwidth. | Requires less bandwidth. |
| Less complex configuration. | More complex configuration. |
| Sufficient for small internetworks. | Sufficient for large internetworks. |

Now that you have a better idea of the pros and cons of the various routing protocols, here are some possible scenarios and their solutions.

**Scenario and Solution**

| | |
|---|---|
| When routers have low memory but are connected to fast links. | Use a distance-vector routing protocol. |
| When routers have an abundance of memory but are connected to slow links. | Use link-state routing protocols. |

## 12.5 Congestion Control

Congestion is said to occur in the network when the resource demands exceed the capacity and packets are lost due to too much queuing in the network. When routers are receiving packets faster than they can forward them, one of two things must happen:

- The subnet must prevent additional packets from entering the congested region until those already present can be processed.

- The congested routers can discard queued packets to make room for those that are arriving.

During congestion, the network throughput may drop to zero and the path delay may become very high. **A congestion control scheme** helps the network to recover from the congestion state. **A congestion avoidance scheme** allows a network to operate in the region of low delay and high throughput. Such schemes prevent a network from entering the congested state. Congestion avoidance is a prevention mechanism while congestion control is a recovery mechanism.

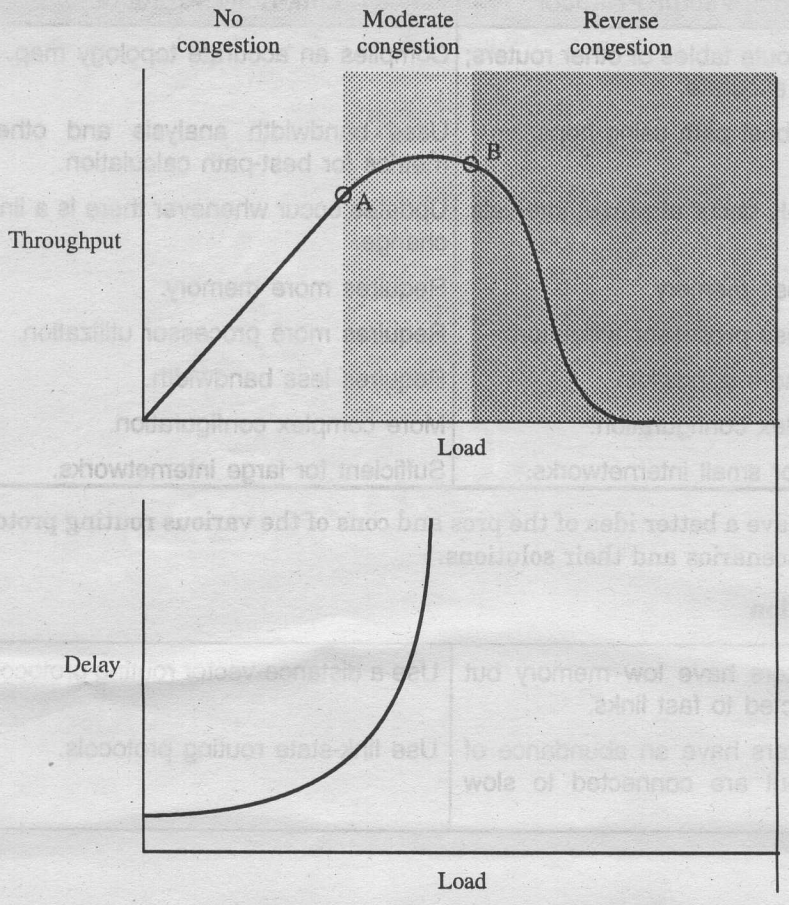



**Fig. 12.13** Graph showing levels of congestion

## 12.5.1 Flow Control vs. Congestion Control

The terms flow control and congestion control are distinct. Flow control is an agreement between a source and a destination to limit the flow of packets without taking into account the load on the network. The purpose of flow control is to ensure that a packet arriving at a destination will find a buffer there. Congestion control is primarily concerned with controlling

the traffic to reduce overload on the network. Flow control is a data link issue, and concerns only one sender outrunning a single receiver (e.g. a point-to-point link). Congestion control is a network layer issue, and is thus concerned with what happens when there is more data in the network than can be sent with reasonable packet delays, no lost packets, etc. Flow control is a local and congestion control is global. Flow control solves the problem of the destination resources being the bottleneck while congestion control solves the problem of the routers and links being the bottleneck. Different connections on a network can choose different flow control strategies, but nodes on the network should follow the same congestion control strategy, if it is to be useful. The two parties in flow control are generally interested in cooperating whereas the n parties (e.g., different users) in congestion control may be non-cooperative. Fairness is not an issue for the two cooperating parties whereas it is an important issue for n competing parties.

From the above discussion the main points of comparison are summarized in the following table:

| | |
|---|---|
| • **Congestion control** is needed when buffers in packet switches overflow or congest. | • **Flow control** is needed when the buffers at the receiver are not depleted as fast as the data arrives. |
| • Congestion is end to end; it includes all hosts, links and routers. | • Flow is between one data sender and one receiver. It can be done on link-to-link or end-to-end basis. |
| • If viewed as a "congested buffer", then the switch tells the source of the data stream to slow down using congestion control notifications. When output buffers at a switch fill up and packets are dropped this leads to congestion control actions. | • If there is a queue on the input side of a switch, and link-by-link flow control is used, then as a "flow control" action the switch tells its immediate neighbor to slow down if the input queue fills up. |

### 12.5.2 Causes of Congestion

The main causes of congestion are stated below:

1. Exhaustion of buffer space
2. Deadlock

**1. Exhaustion of Buffer Space :** Routers maintain packet queues (or buffers). Buffers fill up if either routers are too slow or combined input traffic rate exceeds the outgoing traffic rate. Insufficient buffer space leads to congestion.

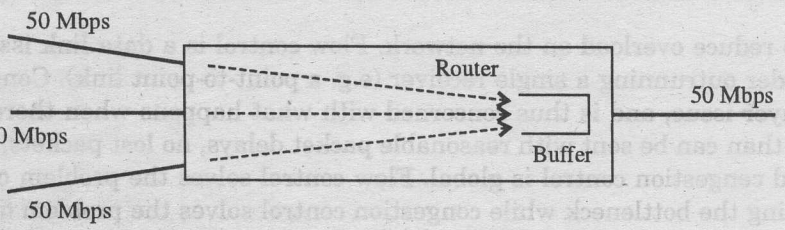



**Fig. 12.14** Exhaustion of buffer space

**2. Deadlock :** It is a state in which the first router cannot proceed until the second router does something, and the second router cannot proceed until the first router does something. Both routers come to a completely halt and stay that way forever

**Types of Deadlock**

1. Store and Forward Lockup

- Direct Store and Forward Lockup
- Indirect Store and Forward Lockup

2. Reassembly Lockup

**Direct Store and Forward Lockup**

It is the simplest lockup between two routers.

***Example*** :

- Suppose router A has five buffers, all of which are queued for output to router B.
- Similarly, router B has five buffers, all of which are queued for output to router A.
- If there is flow control on the link between routers A and B, then neither router can accept any incoming packets from the other. They are both stuck.

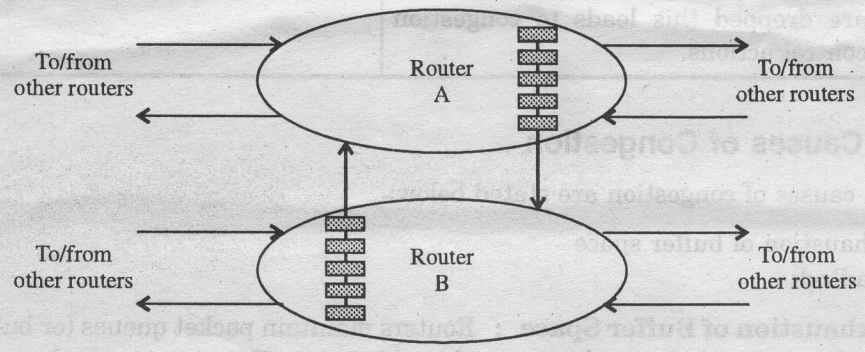



**Fig. 12.15 (a)** Direct store and forward lockup

**Indirect Store and Forward Lockup**

The same thing can happen on a larger scale.

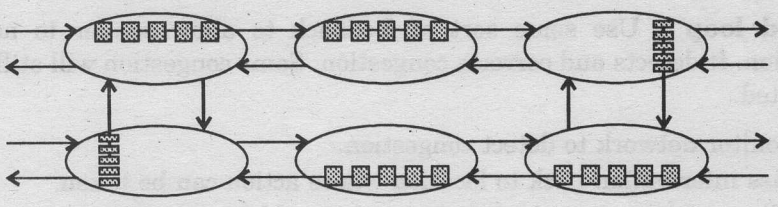


Fig. 12.15 (b) Indirect store and forward lockup

### Reassembly Lockup

In some network layer implementations, the sending router must split messages into multiple network layer packets. The receiving routers reassemble split up packets into a single packet. If the receiving router's buffer fills up with incomplete packets, it cannot reassemble any more packets.

## Congestion Control

We need congestion control so that the application can better achieve its goals of minimizing loss and delay, maximizing throughput. Congestion control algorithms are used to control the number of packets delivered in the network where the traffic is on the boundary of the network carrying capacity.

### Desirable properties of congestion control

- Congestion control should result in many packets delivered at short delays.
- Protect network from congestive collapse but still transport as much data as possible.
- *Fairness* : Give all participating flows a "fair" share of available capacity.

### Design options for congestion control mechanisms

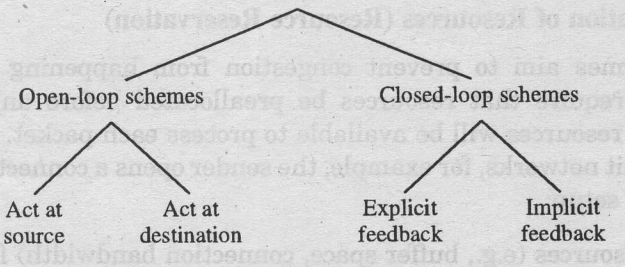



Fig. 12.16 Congestion control mechanisms

- **Open loop** : Design system up front so that it will work correct, no corrections at runtime necessary. It prevents congestion from every happening. It tends to be conservative and result in under utilization.
  - ☐ Request resources before using them.
  - ☐ Global (or regional) resource allocation.
  - ☐ Responds yes or no to each request for service.

- **Closed loop :** Use some sort of feedback to allow sender to adapt to current situation. It detects and corrects congestion. Some congestion will still occur until it is corrected.
  - Monitor network to detect congestion.
  - Pass information back to location where action can be taken.
  - Adjust system operation to correct the problem.
- **Explicit feedback :** Point where congestion occurs informs sender.
- **Implicit feedback :** No explicit action taken; congestion is deduced by sender from the network's behavior (e.g., missing acknowledgements)

### 12.5.3 Types of Congestion Control

There are two techniques for congestion control:

**1. Preventive :** The hosts and routers attempt to prevent congestion before it can occur.

**2. Reactive :** The hosts and routers respond to congestion after it occurs and then attempt to stop it.

**Preventive Techniques:**

- Resource reservation/Preallocation of resources
- Traffic shaping - Leaky/Token bucket
- Isarithmic control

**Reactive Techniques:**

- Load shedding
- Choke packets

#### 12.5.3.1 Preallocation of Resources (Resource Reservation)

Preallocation schemes aim to prevent congestion from happening in the first place. For example, we can require that resources be preallocated before any packets can be sent, guaranteeing that resources will be available to process each packet.

In virtual circuit networks, for example, the sender opens a connection before sending data. During connection setup:

- Request resources (e.g., buffer space, connection bandwidth) from the network.
- If the network has enough available resources to support the new connection, the connection will be established.
- Otherwise, the connection will be rejected.

When the subnet is congested, it can refuse to open the connection, forcing the user to wait until sufficient resources become available.

**Note:** The ability of the subnet to reject requests to open connections is an important property of connection oriented networks

**Resource Reservation: Example**

**Case 1 :** Source attempts to connect to destination, and attempts to reserve 4 Mbps for the connection.

**Result :** Connection accepted. There is enough bandwidth available. Available link bandwidths updated.

**Case 2 :** Source attempts to connect to destination, and attempts to reserve 5 Mbps for the connection.

**Result :** Failure. There is not enough bandwidth available on one of the links.

### 12.5.3.2 Packet Discarding/Load Shedding

Packet discarding operates at the other end of the scale to preallocation of buffers.

If a packet arrives at one of the intermediate IMPs and there are no free buffers, then the packet will just be thrown away. Obviously it would be ridiculous to simply discard at will any packets for which there is no room, as they could potentially contain information which could lead to buffers becoming free. With this in mind, one buffer on each line can be permanently reserved to accept any incoming packet and look at its contents to see if it would be useful. Its contents can then be utilized or the packet discarded. Either a new buffer will become the 'customs' buffer or the old one will be cleared ready for the next packet to arrive and be checked. In this way, packets are not thrown away to the detriment of the network.

**Intelligent Load Shedding**

- Discarding packets does not need to be done randomly.
- Router should take other information into account. Possibilities:
  - Total packet dropping
  - Priority discarding
  - Age biased discarding

**Total Packet Dropping**

- When the buffer fills and a packet segment is dropped, drop all the rest of the segments from that packet, since they will be useless anyway.
- Only works with routers that segment and reassemble packets.

**Priority Discarding**

- Sources specify the priority of their packets.
- When a packet is discarded, the router chooses a low priority packet.
- Requires hosts to participate by labeling their packets with priority levels.

**Age Biased Discarding**

- When the router has to discard a packet, it chooses the oldest one in its buffer.
- This works well for multimedia traffic that requires short delays.

- This may not work so well for data traffic, since more packets will need to be retransmitted.

### 12.5.3.3 Isarithmic Congestion Control

Another approach to congestion avoidance is to limit the total number flow of packets in the subnet at any one time. This method attempts to tackle the fundamental reason for congestion within the subnet. Proposed by Davies in 1972, this method says that if congestion is caused by there being too many packets in the subnet, then it can be solved by setting an upper limit on the number allowed to be present at any given time.

The idea is similar to the token ring:

1. When a router accepts a packet from a host, it must obtain a permit before sending the packet into the subnet.
2. Obtaining a permit is analogous to "seizing the token", but there can be many permits in the subnet. When a router obtains a permit, it destroys it.
3. The destination router regenerates the permit when it passes the packet to the destination host.

The isarithmic congestion control method is however a long way from being ideal. Firstly the method does nothing about the fact that individual IMPs can become congested. Even though there are a limited number of packets allowed in the subnet, all of these could conceivably be going to the same IMP, causing congestion. Also, the procedure for dealing with the permits themselves is not very good. A fair way of distributing the permits throughout the subnet, and also allowing for a situation where a permit is accidentally destroyed, are both very difficult to do efficiently and effectively.

Thus, permit management and individual congestion problems at IMPs mean that the isarithmic congestion control method is not a suitable solution to the problem.

**Isarithmic Control—Example**

- Each host is initially allocated a pool of permits.

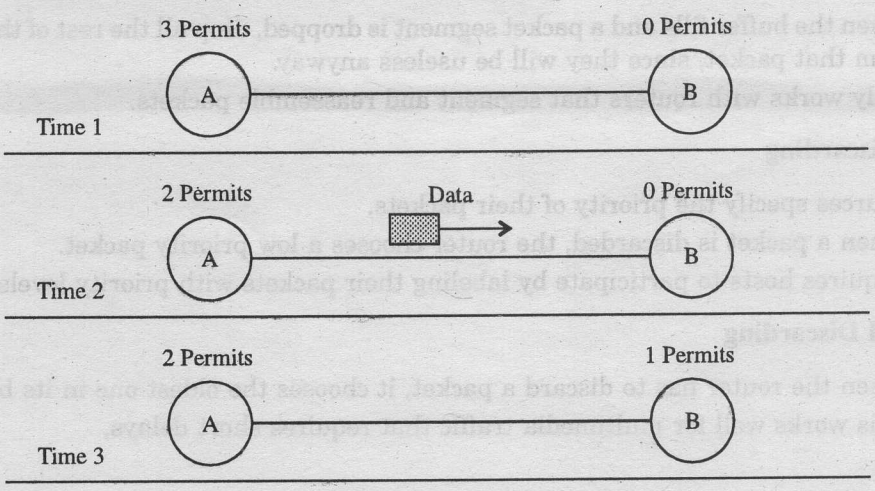



**Fig. 12.17** Example showing isarithmic control

- Data packets carry "permits" between hosts.
- A host can't send a packet unless it has at least one permit.
- If no permit is available, the packet waits in the host until a permit becomes available.
- Isarithmic control is implemented at the hosts, not the routers
- Isarithmic control guarantees that the number of packets in the network will never exceed the number of permits initially present

■ **Problem :**

*If a host doesn't send any packet, i.e. it just holds onto its permits, then the network will not be utilized fully.*

**Solution :** Place a limit on the number of permits a host may keep. If the host has more than its allowed number of permits, it must transmit the excess permits to other hosts by piggybacking them onto other data packets or by using special permit-transport packets

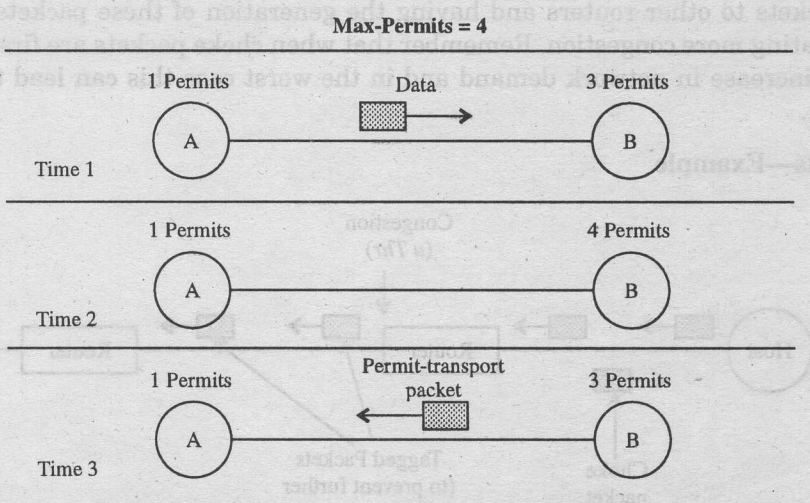



**Fig. 12.18** A case of isarithmic control with max-permits = 4

**Disadvantages of Isarithmic Congestion Control**

1. Although we have limited the total number of packets in the subnet, we have no control over where in the subnet those packets will be. Thus, a router in one part of the subnet might be congested, while a router in another part remains idle, but unable to process packets for lack of permits.
2. Regenerating lost permits is difficult, because no single node knows how many permits are currently in the subnet.
3. How to distribute permits? Distribute among the routers or centralize for a known access point.

### 12.5.3.4 Choke Packets

A better method is to have each IMP monitor the level of traffic on its output lines. If this is below a certain level then everything is fine, but if the traffic goes above this level then the

IMP in question generates a choke packet containing the destination address of the offending packet. The source then retransmits that packet with a flag set so that no more choke packets will be generated because of it. It then slows down the rate at which it is currently sending. Any packets already on the move will be allowed to continue.

If the congestion clears, then the source can move back up to its normal speed.

How can a router measure congestion? A router might estimate the level of congestion by measuring the percentage of buffers in use, line utilizations, or average queue lengths.

The choke packet is sent from a router to all the sources of data associated with the congestion. The sources respond by reducing the flow by about a factor of two, which would mostly remove the congestion. However, this is not guaranteed as there can always be new hosts trying to transmit and these can have the effect of constantly bringing the congestion back above threshold. A source trying to complete a long transfer can be strongly hindered by lots of sources completing small transfers.

The reason for marking the packet as a choke packet is that we do not want routers sending out choke packets to other routers and having the generation of these packets spiral out of control by creating more congestion. Remember that when choke packets are first created they represent an increase in network demand and in the worst case this can lead to a reduction in throughput.

**Choke Packets—Example**

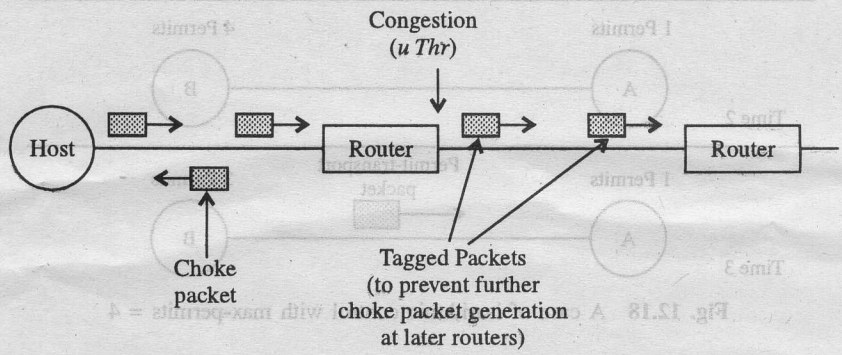



**Fig. 12.19** Example of choke packets

- Each router monitors the utilization of each of its output lines.
- Associated with each line is a variable u, which reflects the utilization of that line.
- Whenever u moves above a given threshold value, the output line enters a "warning state".
- Each newly arriving packet checks if its output line is in the warning state.
- If so, the router sends a choke packet back to the source.
- The data packet is tagged (by setting a bit in its header) so that it will not generate any more choke packets at downstream routers.
- When the source host receives the choke packet, it is required to reduce its traffic generation rate to the specified destination by X%.

- Since other packets aimed at the same destination are probably already on their way to the congested location, the source host should ignore choke packets for that destination for a fixed time interval. After that, it resumes its response to choke packets.

**Advantages:**

Dynamic. Host sends as much data as it wants, the network informs it when it is sending too much.

**Disadvantages:**

1. Difficult to tune. By how much should a host slow down? The answer depends on how much traffic the host is sending, how much of the congestion it is responsible for, and the total capacity of the congested region. Such information is not readily available in practice.
2. After receiving a choke packet, the sending host should ignore additional choke packets for a short while because packets currently in transmission may generate additional choke packets. How long? Depends on such dynamic network conditions as delay.
3. Another problem stems from the global scale of networks. It can take a long time for a choke packet to reach the source. If the source is a fast machine it may have already sent a huge amount of data into the network.

### 12.5.3.5 Traffic Shaping

One of the causes of congestion is the inherent burstiness of computer network traffic. Traffic shaping means to smooth, or otherwise alter, the offered traffic as a function of time. Smooth traffic is easier to deal with the bursty traffic. A subnet may be able to handle on average 10 M packets in an hour, but it probably can't handle 10M packets in one minute and nothing for the next 59 minutes. Traffic shaping minimizes the occurrence of traffic bursts on the network. You can shape traffic by segmenting it, placing it into buffers, and delaying its propagation into the network. These actions ensure a more constant flow of network traffic.

Traffic shaping is performed at all packet interfaces (see the incoming packet traffic in Figure 12.20).

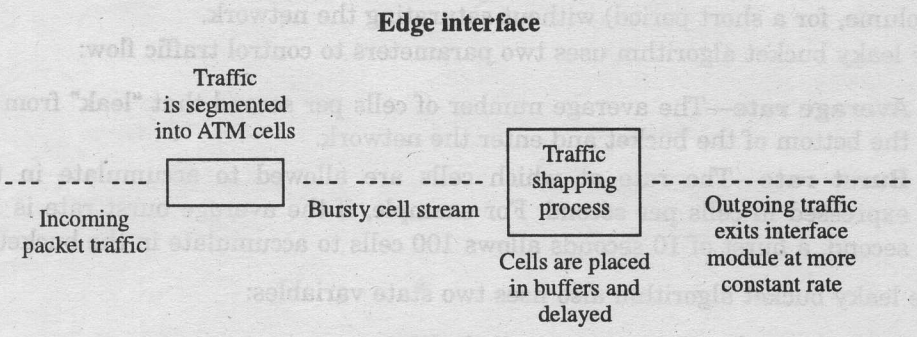



**Fig. 12.20** Traffic Shaping

Related to traffic shaping is the whole idea of agreeing with a network service provider to a flow specification. A flow specification describes what sort of traffic you will put into the net, and what sort of quality if service you expect from it. Some of the qualities of service parameters are:

- Loss sensitivity (number of lost bytes per unit of time).
- Loss interval (the unit of time for calculating loss).
- Burst loss sensitivity (how long of a loss burst can be tolerated).
- Minimum delay (how large of a delay before the application notices).
- Maximum delay variation (the variance or jitter in the inter-packet delay).
- Quality of guarantee (how serious the spec is to the application).

The two algorithms for traffic shaping are:

- Leaky Bucket Algorithm.
- Token Bucket Algorithm.

## Leaky Bucket Algorithm

It is basically a buffer that converts an unregulated, bursty traffic flow into a regulated, smooth, predictable flow. You put the buffer in between a traffic source and the subnet. The buffer acts like a single server queue with a finite queue length. Packets put in the buffer when it is full are thrown away. The buffer may drain onto the subnet either by some number of packets per unit time, or by some total number of bytes per unit time (helpful if packets vary greatly in size).

The leaky bucket algorithm behaves like a bucket with a hole in the bottom. Water can only leak from the bucket at a fixed rate. If data flows into the bucket faster than data flows out through the hole, the bucket eventually "overflows", causing data to be discarded until enough volume again exists in the bucket to accept new data. It provides a single server queueing model with a constant service time.

If packets are of fixed sizes (such as 53-byte ATM cells), one packet may be transmitted per unit time. If packets are of variable size, a fixed number of bytes (perhaps multiple packets) may be admitted. The leaky bucket algorithm enables an application to generate bursty traffic (high volume, for a short period) without saturating the network.

The leaky bucket algorithm uses two parameters to control traffic flow:

- **Average rate**—The average number of cells per second that "leak" from the hole in the bottom of the bucket and enter the network.
- **Burst rate**—The rate at which cells are allowed to accumulate in the bucket, expressed in cells per second. For example, if the average burst rate is 10 cells per second, a burst of 10 seconds allows 100 cells to accumulate in the bucket.

The leaky bucket algorithm also uses two state variables:

- **Current time**—The current wall clock time.

- **Virtual time**—A measure of how much data has accumulated in the bucket, expressed in seconds.

For example, if the average rate is 10 cells per second and 100 cells have accumulated in the bucket, then the virtual time is 10 seconds ahead of the current time.

**Features of Leaky Bucket**

It is a single-server queueing system with constant service time. Following are the main features of this system:

- Used in conjunction with resource reservation to police the host's reservation.
- The leaky bucket is a "traffic shaper": It changes the characteristics of packet stream.
- Each host is connected to the network by an interface containing leaky bucket.
- If a packet arrives at the queue when it is full, the packet is discarded, otherwise is appended to the end of the queue.
- The host is allowed to put one packet per clock tick from "leaky bucket" onto the network. (Enforced either by the interface card or by the operating system).
- Usually the network tells the leaky bucket the rate at which it may send packets when the connection begins.

**Leaky Bucket: Analogy**

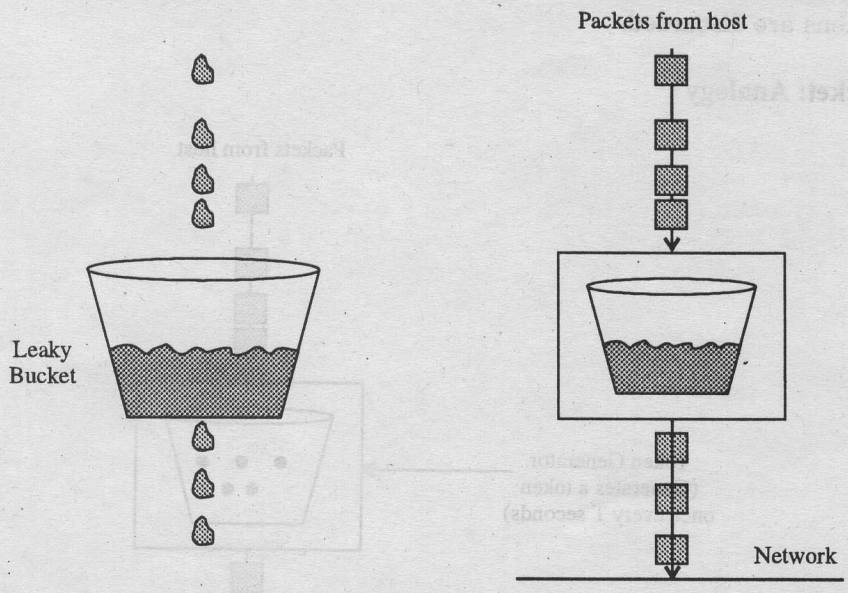



**Fig. 12.21** Leaky bucket

**Leaky Bucket doesn't allow bursty transmissions.** In some cases, we may want to allow short bursts of packets to enter the network without smoothing them out. For this purpose we use a token bucket, which is a modified leaky bucket

## Token Bucket Algorithm

The leaky bucket algorithm enforces a strict maximum traffic generation. It is often better to permit short bursts, but thereafter to constrain the traffic to some maximum. A variant on the leaky bucket is the token bucket. The bucket is filled with tokens at a certain rate. A packet must grab and destroy a token to leave the bucket. Packets are never lost; they just have to wait for an available token. Some bursts are allowed (up to the number tokens in the bucket) which is a good match for some applications (compromise with congestion and source needs).

Packets are placed in an "infinite" queue. Packets may enter the subnet whenever a token is available, else they must wait. The token bucket algorithm enables a host to transmit in short bursts, effectively "saving up" limited permission to generate a burst. The implementation of token bucket is just a counter (variable) that counts tokens: incremented by one every tick and decremented by one whenever packet is sent. When the counter hits 0, no packets can be sent.

**Following are the main features of token bucket:**

- The bucket holds tokens instead of packets.
- Tokens are generated and placed into the token bucket at a constant rate.
- When a packet arrives at the token bucket, it is transmitted if there is a token available. Otherwise it is buffered until a token becomes available.
- The token bucket has a fixed size, so when it becomes full, subsequently generated tokens are discarded.

**Token Bucket: Analogy**

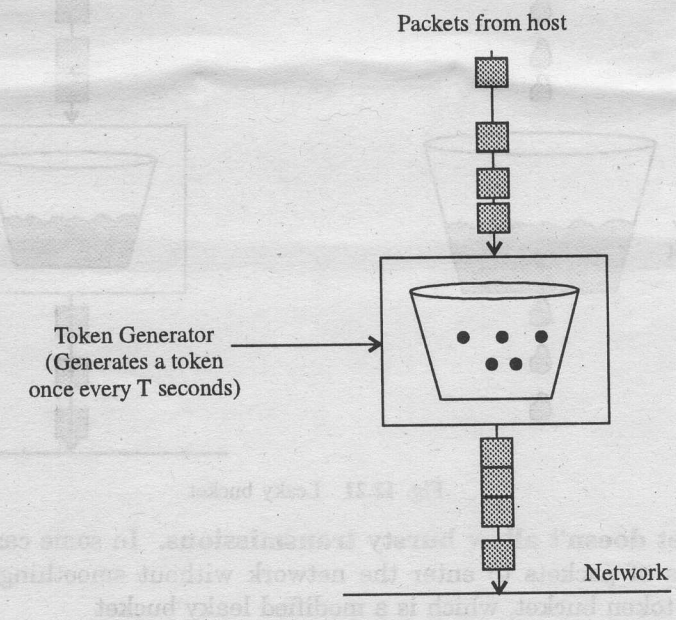



**Fig. 12.22** Token bucket

**Another way to illustrate token bucket:**

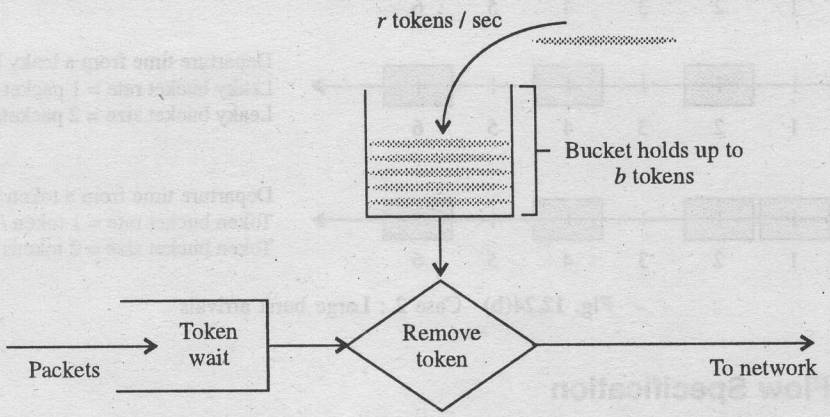



**Fig. 12.23** Illustration of token bucket algorithm

**Token Bucket vs. Leaky Bucket**

| Leaky bucket | Token bucket |
|---|---|
| • Leaky Bucket (LB) discards packets. | • Token Bucket (TB) discards tokens. |
| • With LB, a packet can be transmitted if the bucket is not full. | • With TB, a packet can only be transmitted if there are enough tokens to cover its length in bytes. |
| • LB sends the packets at an average rate. | • TB allows for large bursts to be sent faster by speeding up the output. |
| • LB does not allow saving, a constant rate is maintained. | • TB allows saving up tokens (permissions) to send large bursts. |

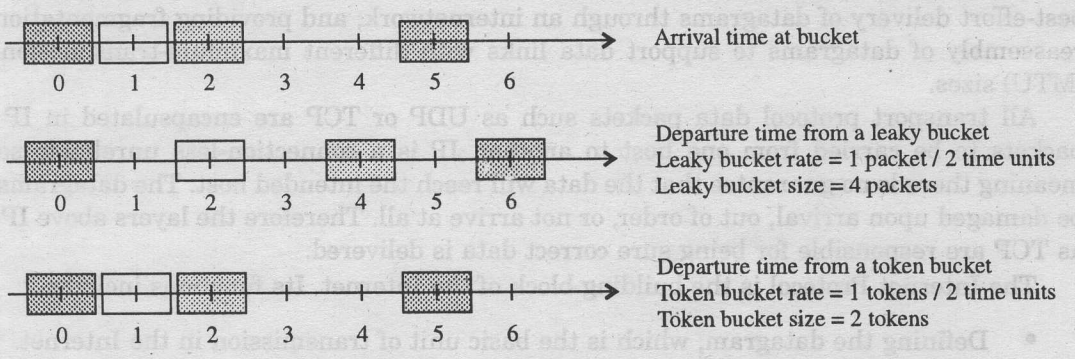



**Fig. 12.24(a)** Case 1 : Short burst arrivals

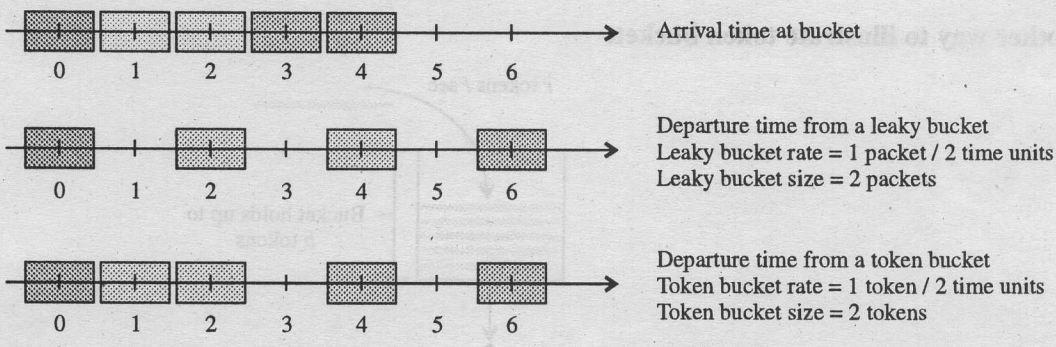



**Fig. 12.24(b)** Case 2 : Large burst arrivals

### 12.5.3.6 Flow Specification

It is also like traffic shaping open loop method applicable both to virtual circuits and datagrams. The sender, receiver, and subnet agree on flow specification:

- The maximum transmission rate.
- Maximum packet size.
- Token bucket rate and size.
- Minimum delay noticed.
- Maximum delay variation (jitter).
- Burst loss sensitivity (how many consecutive lost packets can be tolerated).

## 12.6 The IP Protocol

The Internet Protocol (IP) is a network-layer (Layer 3) protocol that contains addressing information and some control information that enables packets to be routed. IP represents the heart of the Internet protocols. IP has two primary responsibilities: providing connectionless, best-effort delivery of datagrams through an internetwork; and providing fragmentation and reassembly of datagrams to support data links with different maximum-transmission unit (MTU) sizes.

All transport protocol data packets such as UDP or TCP are encapsulated in IP data packets to be carried from one host to another. IP is a connection-less unreliable service meaning there is no guarantee that the data will reach the intended host. The datagrams may be damaged upon arrival, out of order, or not arrive at all. Therefore the layers above IP such as TCP are responsible for being sure correct data is delivered.

The Internet Protocol is the building block of the Internet. Its functions include:

- Defining the datagram, which is the basic unit of transmission in the Internet.
- Defining the Internet addressing scheme.
- Moving data between the Network Access Layer and the Host-to-Host Transport Layer.

- Routing datagrams to remote hosts.
- Performing fragmentation and re-assembly of datagrams.

### 12.6.1 Characteristics of IP

- First, IP is a **connectionless protocol**. This means that IP does not exchange control information (called a "handshake") to establish an end-to-end connection before transmitting data. In contrast, a connection-oriented protocol exchanges control information with the remote system to verify that it is ready to receive data before any data is sent. When the handshaking is successful, the systems are said to have established a connection. Internet Protocol relies on protocols in other layers to establish the connection if they require connection-oriented service.
- IP also relies on protocols in the other layers to provide error detection and error recovery. The Internet Protocol is sometimes called an **unreliable protocol** because it contains no error detection and recovery code. This is not to say that the protocol cannot be relied on - quite the contrary. IP can be relied upon to accurately deliver your data to the connected network, but it doesn't check whether that data was correctly received. Protocols in other layers of the TCP/IP architecture provide this checking when it is required.

### 12.6.2 IP Packet Format

The Internet Protocol (IP) uses a Datagram service to transfer packets of data between end systems using routers. An IP packet contains several types of information, as illustrated in figure 12.25.

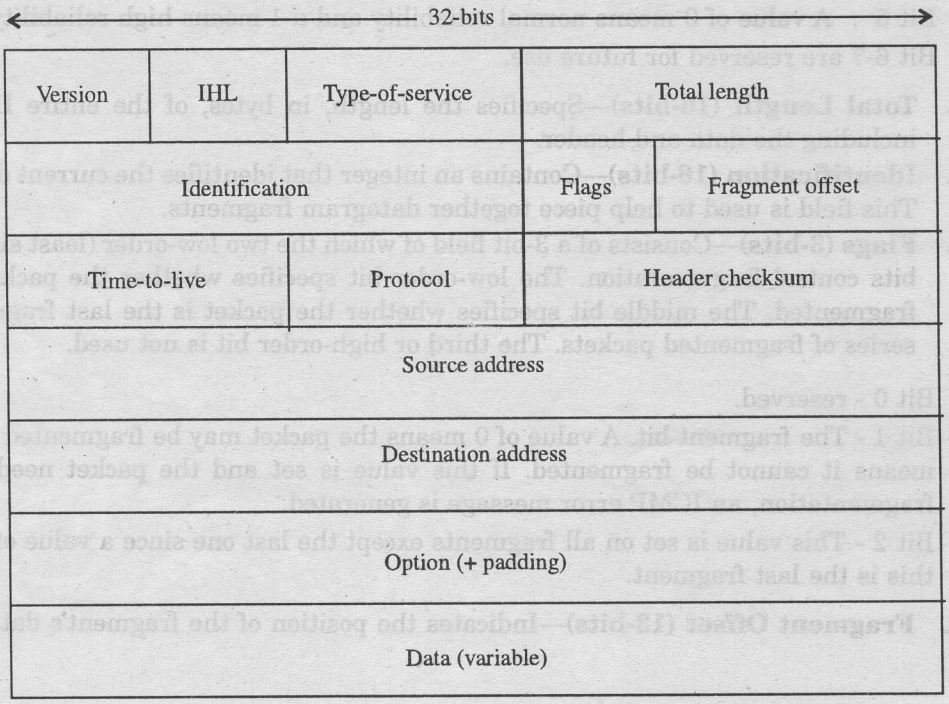



**Fig. 12.25** IP packet format

The various fields are described below:

1. **Version (4 bits)**—Indicates the IP protocol version, currently 4 or 6.
2. **IP Header Length (IHL) (4 bits)**—Indicates the datagram header length in 32-bit words.
3. **Type-of-Service (8 bits)**—Specifies how an upper-layer protocol would like a current datagram to be handled, and assigns datagrams various levels of importance. Only 4 bits are used which are minimize delay, maximize throughput, maximize reliability, and minimize monetary cost. Only one of these bits can be on. If all bits are off, the service is normal. Some networks allow a set precedence to control priority of messages; the bits are as follows:

- Bits 0-2: Precedence.
  - 111 - Network Control.
  - 110 - Internetwork Control.
  - 101 - CRITIC/ECP.
  - 100 - Flash Override.
  - 011 - Flash.
  - 010 - Immediate.
  - 001 - Priority.
  - 000 - Routine.
- Bit 3 : A value of 0 means normal delay. A value of 1 means low delay.
- Bit 4 : Sets throughput. A value of 0 means normal and a 1 means high throughput.
- Bit 5 : A value of 0 means normal reliability and a 1 means high reliability.
- Bit 6-7 are reserved for future use.

4. **Total Length (16-bits)**—Specifies the length, in bytes, of the entire IP packet, including the data and header.
5. **Identification (16-bits)**—Contains an integer that identifies the current datagram. This field is used to help piece together datagram fragments.
6. **Flags (3-bits)**—Consists of a 3-bit field of which the two low-order (least significant) bits control fragmentation. The low-order bit specifies whether the packet can be fragmented. The middle bit specifies whether the packet is the last fragment in a series of fragmented packets. The third or high-order bit is not used.

- Bit 0 - reserved.
- Bit 1 - The fragment bit. A value of 0 means the packet may be fragmented while a 1 means it cannot be fragmented. If this value is set and the packet needs further fragmentation, an ICMP error message is generated.
- Bit 2 - This value is set on all fragments except the last one since a value of 0 means this is the last fragment.

7. **Fragment Offset (13-bits)**—Indicates the position of the fragment's data relative

to the beginning of the data in the original datagram, which allows the destination IP process to properly reconstruct the original datagram.

8. **Time-to-Live (8-bits)**—Maintains a counter that gradually decrements down to zero, at which point the datagram is discarded. This keeps packets from looping endlessly.
9. **Protocol (8-bits)**—Indicates which upper-layer protocol receives incoming packets after IP processing is complete. This may be one or more of TCP, UDP, ICMP, IGMP or OSPF.
10. **Header Checksum (16-bits)**—Helps ensure IP header integrity.
11. **Source Address (32-bits)**—Specifies the sending node.
12. **Destination Address (32-bits)**—Specifies the receiving node.
13. **Options**—Allows IP to support various options, such as security. The various options are listed below:

Security and handling restrictions.

- Record route—Each router records its IP address.
- Time stamp—Each router records its IP address and time.
- Loose source routing—Specifies a set of IP addresses the datagram must go through.
- **Strict source routing**—The datagram can go through only the IP addresses specified.

14. **Data**—Contains upper-layer information.

Fragmentation is handled at the IP network layer and the messages are reassembled when they reach their final destination. If one fragment of a datagram is lost, the entire datagram must be retransmitted. This is why fragmentation is avoided by TCP. The data on the last line, item 14, is Ethernet data, or data depending on the type of physical network.

### 12.6.3 IP Addressing

As with any other network-layer protocol, the IP addressing scheme is integral to the process of routing IP datagrams through an internetwork. Each IP address has specific components and follows a basic format. These IP addresses can be subdivided and used to create addresses for subnetworks.

An address is a data structure understood by a network, which uniquely identifies the recipient within the network. Addresses in other places than computer networks: Addresses are used by the postal system to allow a postman to find a person's house; to allow a computer to uniquely identify a location in memory.

A unicast/broadcast IP address is a 32-bit value (i.e., four bytes), which is allocated to each system in the Internet. The 32-bit value uniquely identifies this system, and therefore no two systems may have the same IP address. Some systems have more than one IP address, in which case they may be reached by any of their IP addresses.

Each IP address consists of two parts, the network part (identifying the network number, or LAN collision domain, to which the computer is attached) and the host part (which identifies the host within the local network). IP addresses are normally written in a format known as

"dotted decimal notation". In this format, each byte of the 4-byte address is expressed as a decimal (base 10) number (i.e., 0 to 255). The four decimal numbers are separated by "dots" or "periods" as shown below:

IP address "129.7.1.10" corresponds to a hexadecimal value of $0 \times 8107010A$.

### IP Address Format

The 32-bit IP address is grouped eight bits at a time, separated by dots, and represented in decimal format (known as dotted decimal notation). Each bit in the octet has a binary weight (128, 64, 32, 16, 8, 4, 2, 1). The minimum value for an octet is 0, and the maximum value for an octet is 255. The figure 12.26 illustrates the basic format of an IP address.

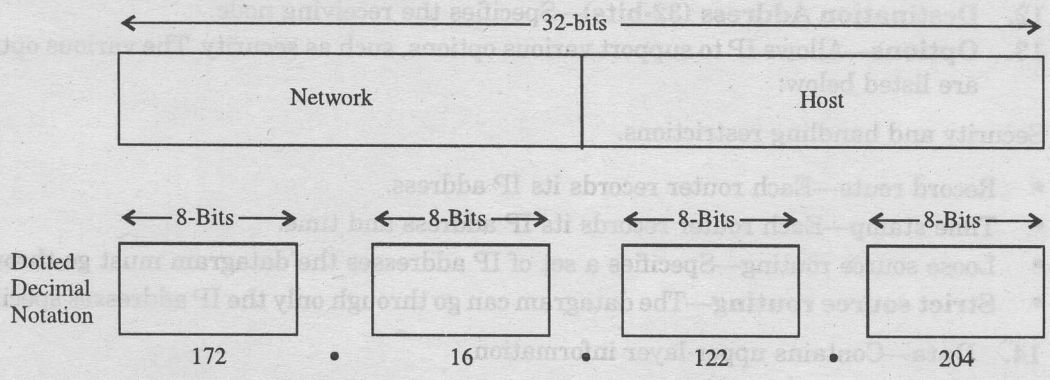



**Fig. 12.26** An IP address consists of 32-bits, grouped into four octets

### IP Subnet Addressing

IP networks can be divided into smaller networks called subnetworks (or subnets). Subnetting provides the network administrator with several benefits, including extra flexibility, more efficient use of network addresses and the capability to contain broadcast traffic (a broadcast will not cross a router).

Subnets are under local administration. As such, the outside world sees an organization as a single network and has no detailed knowledge of the organization's internal structure.

A given network address can be broken up into many subnetworks. For example, 172.16.1.0, 172.16.2.0, 172.16.3.0 and 172.16.4.0 are all subnets within network 171.16.0.0. (All 0s in the host portion of an address specifies the entire network.)

## 12.7 An Introduction To IPv6 : The New Internet Protocol

IPv6 is short for "Internet Protocol Version 6". IPv6 is designed by the IETF (Internet Engineering Task Force) to replace the current version Internet Protocol, IP Version 4 ("IPv4"). It is also referred to as the Next Generation Internet Protocol (IPng).

Most of today's Internet uses IPv4, which is now nearly twenty years old. IPv4 has been remarkably resilient in spite of its age, but it is beginning to have problems. Most importantly, there is a growing shortage of IPv4 addresses, which are needed by all new machines added

to the Internet. The main issue surrounding IPv6 is addressing-or, the lack of addressing—because many experts believe that we are nearly out of the four billion addresses available in IPv4. Although this seems like a very large number of addresses, multiple large blocks are given to government agencies and large organizations. IPv6 could be the solution to many problems, but it is still not fully developed and is not a standard—yet!

IPv6 fixes a number of problems in IPv4, such as the limited number of available IPv4 addresses. It also adds many improvements to IPv4 in areas such as routing and network auto configuration. IPv6 is expected to gradually replace IPv4, with the two coexisting for a number of years during a transition period. At some point in the next three to seven years the Internet will require a deployed new version of the Internet protocol.

Two factors are driving this: routing and addressing. Global Internet routing based on the on 32-bit addresses of IPv4 is becoming increasingly strained. IPv4 addresses do not provide enough flexibility to construct efficient hierarchies that can be aggregated. Even if the IPv4 routing can be scaled to support a full IPv4 Internet, the Internet will eventually run out of network numbers. There is no question that an IPv6 is needed, but only a question of when.

### 12.7.1 Why IPv6?

There are a number of reasons why IPv6 is appropriate for the next generation of the Internet Protocol. It solves the Internet scaling problem, provides a flexible transition mechanism for the current Internet and was designed to meet the needs of new markets such as nomadic personal computing devices, networked entertainment and device control. It does this in an evolutionary way that reduces the risk of architectural problems.

IPv6 supports large hierarchical addresses that will allow the Internet to continue to grow and provide new routing capabilities not built into IPv4. It has anycast addresses that can be used for policy route selection and has scoped multicast addresses, which provide improved scalability over IPv4 multicast. It also has local use address mechanisms, which provide the ability for "plug and play" installation.

IPv6 provides a platform for new Internet functionality. This includes support for real-time flows, provider selection, host mobility, end-to-end security, auto-configuration, and auto-reconfiguration.

IPv6 is just a new version of IP. It can be installed as a normal software upgrade in Internet devices. It is interoperable with the current IPv4. IPv6 is designed to run well on high performance networks (e.g., ATM) and at the same time is still efficient for low bandwidth networks (e.g., wireless). In addition, it provides a platform for new Internet functionality that will be required in the near future.

Another positive outcome of IPv6 will be better Internet routing using QoS, Quality of Service, which routes packets based on priority. So for example, if one person is pinging a server and another is downloading a file, the one pinging will have less priority in their data transmission than the one downloading a file because the user who is downloading a file has created a data stream that will automatically gain more priority over the simple ICMP data packets.

Lastly, routing will be simplified because the IPv6 information header on each packet is far more flexible and can contain more detailed information than an IPv4 header thus allowing for faster routing of data across a network or the Internet.

IPv6 was designed to take an evolutionary step from IPv4. The changes from IPv4 to IPv6 fall primarily into the following categories:

- Expanded Routing and Addressing Capabilities : IPv6 increases the IP address size from 32-bits to 128-bits, to support more levels of addressing hierarchy and a much greater number of addressable nodes, and simpler auto-configuration of addresses. The scalability of multicast routing is improved by adding a "scope" field to multicast addresses.
- A new type of address called an "anycast address" is defined; to identify sets of nodes where a packet sent to an anycast address is delivered to one of the nodes. The use of anycast addresses in the IPv6 source route allows nodes to control the path, which their traffic flows.
- Header Format Simplification : Some IPv4 header fields have been dropped or made optional, to reduce the common-case processing cost of packet handling and to keep the bandwidth cost of the IPv6 header as low as possible despite the increased size of the addresses. Even though the IPv6 addresses are four times longer than the IPv4 addresses, the IPv6 header is only twice the size of the IPv4 header.
- Improved Support for Options : Changes in the way IP header options are encoded allows for more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future.
- Quality-of-Service Capabilities : A new capability is added to enable the labeling of packets belonging to particular traffic "flows" for which the sender requests special handling, such as non-default quality of service or "real-time" service.
- Authentication and Privacy Capabilities : IPv6 includes the definition of extensions that provide support for authentication, confidentiality and data integrity.

## 12.7.2 IPv6 Header Format

The IPv6 protocol consists of two parts, the basic IPv6 header and IPv6 extension headers.

**IPv6 header:**

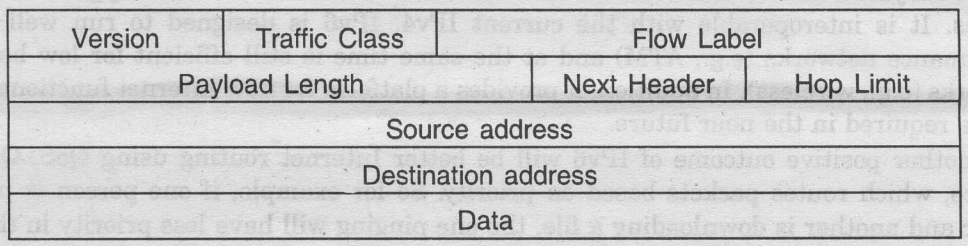



**Fig. 12.27** IPv6 header format

**Version-(4-bits)**—IPv6 version number.

**Traffic Class-(8-bits)**—Internet traffic priority delivery value.

**Flow Label-(20-bits)**—Used for specifying special router handling from source to destination(s) for a sequence of packets.

**Payload Length-(16-bits unsigned)**—Specifies the length of the data in the packet. When cleared to zero, the option is a hop-by-hop Jumbo payload.

**Next Header-(8-bits)**—Specifies the next encapsulated protocol. The values are compatible with those specified for the IPv4 protocol field.

**Hop Limit-(8-bits unsigned)**—For each router that forwards the packet, the hop limit is decremented by 1. When the hop limit field reaches zero, the packet is discarded.

**Source address-(16-bytes)**—The IPv6 address of the sending node.

**Destination address-(16-bytes)**—The IPv6 address of the destination node.

### IPv6 Extensions

IPv6 includes an improved option mechanism over IPv4. IPv6 options are placed in separate extension headers that are located between the IPv6 header and the transport-layer header in a packet. Most IPv6 extension headers are not examined or processed by any router along a packet's delivery path until it arrives at its final destination. This facilitates a major improvement in router performance for packets containing options. In IPv4 the presence of any options requires the router to examine all options.

The other improvement is that unlike IPv4 options, IPv6 extension headers can be of arbitrary length and the total amount of options carried in a packet is not limited to 40 bytes. This feature plus the manner in which they are processed, permits IPv6 options to be used for functions that were not practical in IPv4. A good example of this is the IPv6 Authentication and Security Encapsulation options.

In order to improve the performance when handling subsequent option headers and the transport protocol which follows, IPv6 options are always an integer multiple of 8 octets long, in order to retain this alignment for subsequent headers.

The IPv6 extension headers that are currently defined are:

- Routing : Extended Routing.
- Fragmentation : Fragmentation and Reassembly.
- Authentication : Integrity and Authentication. Security
- Encapsulation : Confidentiality.
- Hop-by-Hop Option : Special options which require hop-by-hop processing.
- Destination Options : Optional information to be examined by the destination node.

## 12.7.3 IPv6 Addressing

To properly understand the addressing scheme of IPv6, we first have to understand the syntax of the actual address. With a standard IPv4 address, it consists of 32 bits broken into 8 bit segments separated by periods. For example:

11111111. 11111111. 11111111. 11111111 (binary)
255 . 255 . 255 . 255 (decimal)

With IPv6, you have a 128 bit address broken down into strings of 16 bits separated by colons. For example:

1111111111111111 : 1111111111111111 : 1111111111111111 : 1111111111111111 :
1111111111111111 : 1111111111111111 : 1111111111111111 : 1111111111111111 (binary)

FFFF : FFFF : FFFF : FFFF : FFFF : FFFF : FFFF : FFFF (Hexadecimal)

Let's look at an example of IPv6 address. The address is an eight-part hex address separated by colons (" : "). Each part n can equal a 16-bit number and is eight parts long, providing a 128-bit address length (16 * 8 = 128),

Addresses are n:n:n:n:n:n:n:n n = 4 digit hexadecimal integer,
16 * 8 = 128 address.

1080:0:0:0:8:800:200C:417A Unicast address
FF01:0:0:0:0:0:0:101 Multicast address

So for example, a 16-bit string such as "0010000111011010" would convert to 21DA in hexadecimal. Hexadecimal is a binary addressing system that takes 4-bits of data and converts it into an alphanumeric value from 0 to F. This chart should give you a little better idea of how the binary numbers, decimal numbers and hexadecimal values all compare to each other.

| Binary | Hexadecimal | Decimal |
|---|---|---|
| 0000 | 0 | 0 |
| 0001 | 1 | 1 |
| 0010 | 2 | 2 |
| 0011 | 3 | 3 |
| 0100 | 4 | 4 |
| 0101 | 5 | 5 |
| 0110 | 6 | 6 |
| 0111 | 7 | 7 |
| 1000 | 8 | 8 |
| 1001 | 9 | 9 |
| 1010 | A | 10 |
| 1011 | B | 11 |
| 1100 | C | 12 |
| 1101 | D | 13 |
| 1110 | E | 14 |
| 1111 | F | 15 |

As you notice, the binary digits are always 4 characters (4 bits). The hexadecimal numbers are all 1 character, and the decimal numbers range from 1-2 characters in size. Since a hexadecimal number can't take up more than 1 space, any number greater than 9 has to be represented with a letter ranging from A-F. The decimal column lists the decimal values of each binary number. Therefore, a 16-bit string of data would need to be represented by 4 characters of hexadecimal data, or as we listed before in our example, something like this: 21DA

### Types of IPv6 Addresses

There are 3 types of addresses in IPv6: Unicast, Multicast, and Anycast.

### Unicast

Unicast is a communication between a single host and a single receiver. Packets sent to a unicast address are delivered to the interface identified by that address, as seen in Figure 12.28.

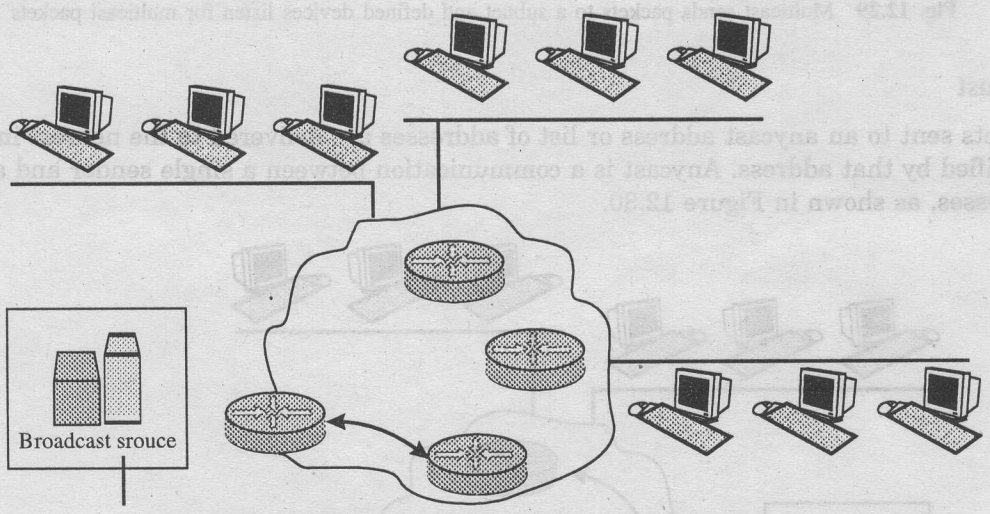



**Fig. 12.28** Unicast sends packets to a specified interface

### Multicast

Multicast is communication between a single host and multiple receivers. Packets are sent to all interfaces identified by that address, as seen in Figure 12.29.

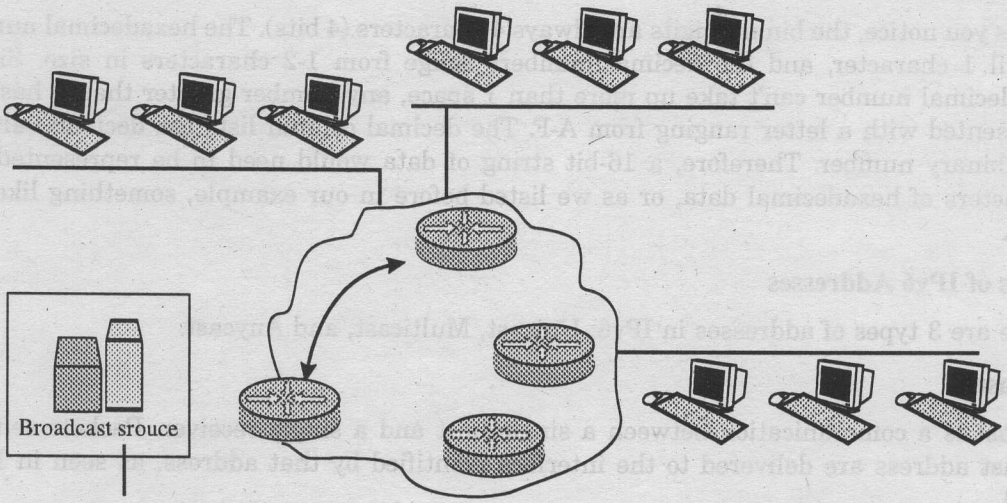



**Fig. 12.29** Multicast sends packets to a subnet and defined devices listen for multicast packets

### Anycast

Packets sent to an anycast address or list of addresses are delivered to the nearest interface identified by that address. Anycast is a communication between a single sender and a list of addresses, as shown in Figure 12.30.

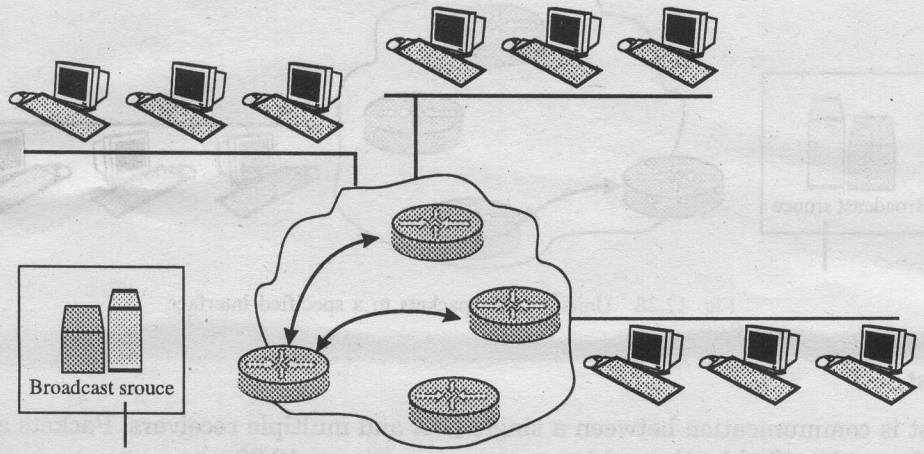



**Fig. 12.30** Anycast sends packets to specified interface list and can contain end nodes and routers

Unicast addresses identify a single interface. Anycast addresses identify a set of interfaces such that a packet sent to an anycast address will be delivered to one member of the set. Multicast addresses identify a group of interfaces, such that a packet sent to a multicast address is delivered to all of the interfaces in the group. There are no broadcast addresses in IPv6, their function being superseded by multicast addresses.

Unicast is standard point A to point B data traffic. This would be nothing more than me

sending data from my computer, across the Internet, to your computer and back again. That's the basic elements of Unicast addressing. So instead of needing to setup a special machine to dictate which server gets a given incoming connection first, IPv6 controls this automatically. Multicast is exactly how it sounds. Sending to one address on a subnet broadcasts to everyone on the subnet. This works the same as the broadcast address in IPv4.

Anycast identifies multiple network interfaces. In the simplest terms, the anycast address system is a "First man wins" system. Basically, if you have three network interfaces identified as part of a given anycast address and a request is sent to the anycast address, it is then forwarded to the first interface in the anycast group. Since that interface is not busy, the packet is then delivered to that interface. Say a second connection comes in and the first interface is busy. The connection is immediately routed over to the second interface. Then a third connection request comes in but now the first interface is not busy anymore so the data is routed to the first interface. This is a simple load balancing system built directly into IPv6.

### Local use Addresses

While many things in IPv6 are new and unique, there are many things that are much the same as they are in IPv4, if not done in a slightly different way. One example of this is local use addresses. These consist of Link Local and Site Local addresses. Link local addresses are more or less the IPv6 equivalent to the MAC address already used on every network out there to identify individual pieces of networking hardware attached to the network. A link local address is automatically configured by the computer and is used for neighbor discovery and for querying of the DHCP server for an IP address and other information necessary on the network. These addresses are not world routable and are only used locally.

Site local addressing would be equivalent to the private network addresses used in Nat under IPv4. These addresses are not world routable and are used for data traffic within your local network itself. Compatibility is also a big factor with IPv6.

## 12.8 Address Resolution Protocol

The address resolution protocol (ARP) is a protocol used by the Internet Protocol (IP), specifically IPv4, to map IP network addresses to the hardware addresses used by a data link protocol. The protocol operates below the network layer as a part of the interface between the OSI network and OSI link layer. It is used when IPv4 is used over Ethernet.

For two machines on a given network to communicate, they must know the other machine's physical (or MAC) addresses. By broadcasting Address Resolution Protocols (ARPs), a host can dynamically discover the MAC-layer address corresponding to a particular IP network-layer address.

The term **address resolution** refers to the process of finding an address of a computer in a network. The address is "resolved" using a protocol in which a piece of information is sent by a client process executing on the local computer to a server process executing on a remote computer. The information received by the server allows the server to uniquely identify the network system for which the address was required and therefore to provide the required

address. The address resolution procedure is completed when the client receives a response from the server containing the required address.

An Ethernet network uses two hardware addresses, which identify the source and destination of each frame, sent by the Ethernet. The destination address (all 1's) may also identify a broadcast packet (to be sent to all connected computers). The hardware address is also known as the Medium Access Control (MAC) address, in reference to the standards, which define Ethernet. Each computer network interface card is allocated a globally unique 6-byte link address when the factory manufactures the card (stored in a PROM). This is the normal link source address used by an interface. A computer sends all packets that it creates with its own hardware source link address, and receives all packets which match the same hardware address in the destination field or one (or more) pre-selected broadcast/multicast addresses.

The Ethernet address is a link layer address and is dependent on the interface card, which is used. IP operates at the network layer and is not concerned with the link addresses of individual nodes, which are to be used. The address resolution protocol (ARP) is therefore used to translate between the two types of address. The ARP client and server processes operate on all computers using IP over Ethernet. The processes are normally implemented as part of the software driver that drives the network interface card.

There are four types of ARP messages that may be sent by the ARP protocol. The types of message are:

1. ARP request.
2. ARP reply.
3. RARP request.
4. RARP reply.

To reduce the number of address resolution requests, a client normally caches resolved addresses for a (short) period of time. The ARP cache is of a finite size, and would become full of incomplete and obsolete entries for computers that are not in use if it was allowed to grow without check. The ARP cache is therefore periodically flushed of all entries. This deletes unused entries and frees space in the cache. It also removes any unsuccessful attempts to contact computers, which are not currently running.

## 12.9 Reverse Address Resolution Protocol (RARP)

RARP, which is the logical inverse of ARP, might be used by diskless workstations that do not know their IP addresses when they boot. RARP relies on the presence of a RARP server with table entries of MAC-layer-to-IP address mappings.

RARP allows a physical machine in a local area network to request its IP address from a gateway server's Address Resolution Protocol (ARP) table or cache. A network administrator creates a table in a local area network's gateway router that maps the physical machine (or Media Access Control—MAC address) addresses to corresponding Internet Protocol addresses (IP address). When a new machine is set up, its RARP client program requests from the RARP server on the router to be sent its IP address. Assuming that an entry has been set up in the

router table, the RARP server will return the IP address to the machine, which can store it for future use.

RARP is available for Ethernet, Fiber Distributed-Data Interface and Token Ring LANs. ARP (Address Resolution Protocol) performs the opposite function as the RARP: mapping of an IP address to a physical machine address.

## 12.10 Internet Control Message Protocol (ICMP)

The Internet Control Message Protocol (ICMP) is a network-layer Internet protocol that provides message packets to report errors and other information regarding IP packet processing back to the source. This protocol is part of the Internet Layer and uses the IP datagram delivery facility to send its messages.

### ICMP Messages

ICMPs generate several kinds of useful messages that are listed below:

- Destination Unreachable.
- Echo Request and Reply.
- Redirect.
- Time Exceeded.
- Router Advertisement.
- Router Solicitation.

If an ICMP message cannot be delivered, no second one is generated. This is to avoid an endless flood of ICMP messages.

When an ICMP **destination-unreachable message** is sent by a router, it means that the router is unable to send the package to its final destination. The router then discards the original packet. Two reasons exist for why a destination might be unreachable. Most commonly, the source host has specified a nonexistent address. Less frequently, the router does not have a route to the destination.

Destination-unreachable messages include four basic types:

- **Network unreachable :** Network-unreachable messages usually mean that a failure has occurred in the routing or addressing of a packet.
- **Host unreachable :** Host-unreachable messages usually indicate delivery failure, such as a wrong subnet mask.
- **Protocol unreachable :** Protocol-unreachable messages generally mean that the destination does not support the upper-layer protocol specified in the packet.
- **Port unreachable :** Port-unreachable messages imply that the TCP socket or port is not available.

An ICMP **echo-request message**, which is generated by the ping command, is sent by any host to test node reachability across an internetwork.

The ICMP **echo-reply message** indicates that the node can be successfully reached.

An ICMP **Redirect message** is sent by the router to the source host to stimulate more efficient routing. The router still forwards the original packet to the destination. ICMP redirects allow host routing tables to remain small because it is necessary to know the address of only one router, even if that router does not provide the best path. Even after receiving an ICMP Redirect message, some devices might continue using the less-efficient route.

An ICMP **Time-exceeded message** is sent by the router if an IP packet's Time-to-Live field (expressed in hops or seconds) reaches zero. The Time-to-Live field prevents packets from continuously circulating the internetwork if the internetwork contains a routing loop. The router then discards the original packet.

**ICMP Router-Discovery Protocol (IDRP)**

IDRP uses Router-Advertisement and Router-Solicitation messages to discover the addresses of routers on directly attached subnets. Each router periodically multicasts Router-Advertisement messages from each of its interfaces. Hosts then discover addresses of routers on directly attached subnets by listening for these messages. Hosts can use Router-Solicitation messages to request immediate advertisements rather than waiting for unsolicited messages.

IRDP offers several advantages over other methods of discovering addresses of neighboring routers. Primarily, it does not require hosts to recognize routing protocols, nor does it require manual configuration by an administrator.

Router-Advertisement messages enable hosts to discover the existence of neighboring routers, but not which router is best to reach a particular destination. If a host uses a poor first-hop router to reach a particular destination, it receives a Redirect message identifying a better choice.

• • •

# 13

# Internetworking Devices

## 13.1 The Concept Of Internetworking

Once networks started becoming popular, it did not take long to realize that connecting many individual networks together would give even greater benefits. It was this realization, which helped the birth of the concept of internetworking that simply refers to the function of connecting individual networks together to form larger networks.

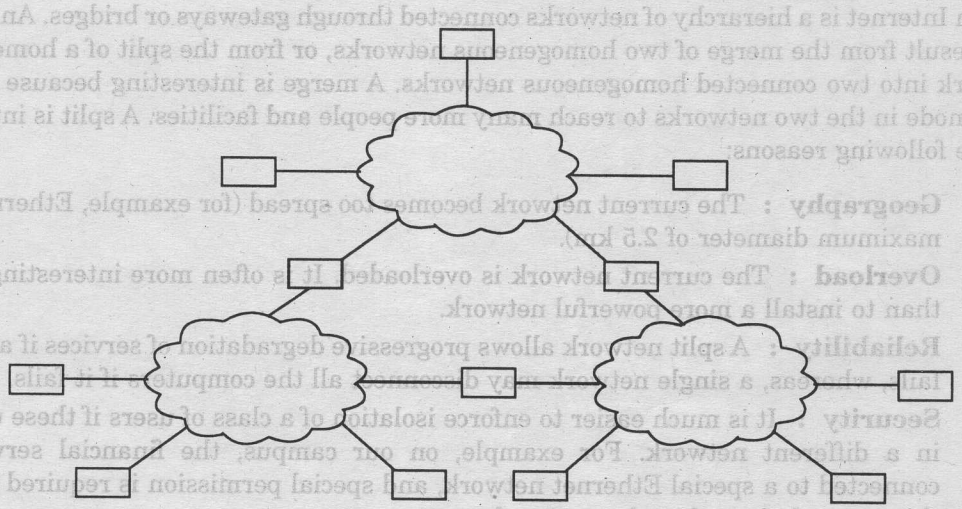


**Fig. 13.1** A Simple internetwork

**Definition**

*The art and science of connecting individual local-area networks (LANs) to create wide-area networks (WANs), and connecting WANs to form even larger WANs. Internetworking can be extremely complex because it generally involves connecting networks that use different protocols. Internetworking is accomplished with routers, bridges, and gateways.*

The premise of internetworking is the connection of two or more networks into an integrated infrastructure capable of allowing communications to occur between the networks. More often, an internetwork is thought of as a network that provides services over the wide area. It should be noted that in all internetworking scenarios, the networks being connected maintain their individuality-they continue to operate as stand-alone networks. The internetworking capabilities are accessed only when the networks need to communicate.

**Principal applications of internetworking are:**

- Fast easy file sharing.
- News broadcast.
- E-Mail.
- Remote logon.
- Messaging gateways.
- Network management.
- Directory services.
- Knowledge discovery.

An Internet is a hierarchy of networks connected through gateways or bridges. An Internet may result from the merge of two homogeneous networks, or from the split of a homogeneous network into two connected homogeneous networks. A merge is interesting because it allows every node in the two networks to reach many more people and facilities. A split is interesting for the following reasons:

- **Geography :** The current network becomes too spread (for example, Ethernet has a maximum diameter of 2.5 km).
- **Overload :** The current network is overloaded. It is often more interesting to split than to install a more powerful network.
- **Reliability :** A split network allows progressive degradation of services if a network fails, whereas, a single network may disconnect all the computers if it fails.
- **Security :** It is much easier to enforce isolation of a class of users if these users are in a different network. For example, on our campus, the financial services are connected to a special Ethernet network, and special permission is required to access this network from the other networks on campus.

### 13.1.1 Why Internetworking?

There are many existing networks located at different places. It is desirable to connect them together for global communication and information exchange. In the future, there will still be multiple types of networks, because no single network technology can fulfill all the needs. It is necessary to connect them for communication.

From the user's point of view it is a set of application programs that use the underlying network to carry out useful communication tasks, e.g., World Wide Web, email, ftp and remote login.

The above applications can run on different types of computers. Different networks may not be compatible. Internetworking requires some additional hardware to interconnect the networks and some additional software on all the attached computers to execute a common set of internetworking protocols to ensure that these computers can communicate. Example: One vendor's email program may talk to another vendor's email program. Different networks may use different technology and methodology, different transmission media, addressing scheme, packet format and etc. No single network hardware can satisfy all constraints of different users/networks.

Internetworking is not simply a plug and play situation. Often there are incompatibility issues that must be dealt with before interoperability can be achieved. It is the consequence of interoperability that is central to internetworking technologies. Interoperability can be divided into two groups of concerns.

- The first is that of actual physical connectivity.
- The second is software-related—concerned with protocols, management and applications.

The two must come together to provide interconnectivity. Interconnectivity requires the use of some type of device to allow interoperability between LAN and WAN environments.

**Features of Internetworks**

Internetworking is more complex than networking, because internetworks must often combine different types of network architectures. Therefore, devices used to create internetworks typically must support:

- Different physical topologies.
- Several Network and Data Link Layer protocols.
- Different physical transmission media.

## 13.2 Introduction To Internetworking Devices

While the concept of Internetworking may seem simple, there is really a lot to be considered before attempting to interconnect a bunch of standalone networks together, assuming you really want the computers to talk to each other. For example, LANs were originally developed with no concept of communicating with other networks. As a result, some additional functionality and equipment need to be added to make this happen. These additional products

which are generally referred to as hubs, repeaters, bridges, routers, switches and gateways - each having a specific task to perform in the overall internetworking scheme of thing.

Internetworking deals with the issues of interconnecting multiple, possibly heterogeneous networks. Physical networks can be connected at several levels. The main problems of internetworking come from the wide variety of types of networks that have to be connected together. Different devices have been built to handle different situations:

- **Repeaters and Hubs :** Repeaters and hubs receive a signal (bits) on a local area network (LAN) segment and regenerate the bit pattern to boost the signal and extend the physical length of the segment. These devices are transparent to the sending and receiving (end) devices. They are not technically internetworking devices, because they only extend the length of a cable, or transmit the same signal to several nodes; however, they are normally thought of as internetworking devices.
- **Bridges and Switches :** Bridges and switches usually connect networks of similar types, such as Ethernet or Token Ring, into a single logical internetwork. Translation bridges have been developed to connect dissimilar LAN types. Bridges and switches store and forward frames to other network segments, and are transparent to the end devices.
- **Routers :** Routers connect networks into internetworks that are physically unified, but in which each network retains its own independent identity. The primary purpose of a router is to find the best path from one network to another network, and forward packets between networks.
- **Gateways :** Gateways operate at higher layers of the protocol stack; some gateways operate on all layers. Gateways, also called protocol converters, connect different networking environments, such as Systems Network Architecture (SNA) and Transmission Control Protocol/Internet Protocol (TCP/IP) networks, by converting each header from the form of the sending environment to that of the receiving environment.
- **Brouters :** A brouter can work in either the second and third layers of the OSI model-the data link layer or the network layer. A brouter is a combination of a bridge and router combined. If it can't route a packet, it acts as a bridge.

**Review of the Open Systems Interconnection Model**

The Table 13.1 provides an overview of the primary functions of each layer of the OSI model, as well as the units of information and devices that correspond to each layer.

| OSI Layer | Layer Function | Unit of Information | Address Type | Internetworking Devices |
|---|---|---|---|---|
| Application | Provide user functionality | Program/ Application Data | | Gateways (Protocol Converters) |
| Presentation | Provide character representation, data compression, security | Characters and words | | Gateways (Protocol Converters) |

| | | | | |
|---|---|---|---|---|
| **Session** | Establish, conduct, and end sessions | | | Gateways (Protocol Converters) |
| **Transport** | Transfer messages between application | Message/Byte Stream | Application process address (port) | Gateways (Protocol Converters) |
| **Network** | Send individual packets across a network | Packet (datagram) | Network address | Routers, Layer 3 Switches |
| **Data Link** | Send frames to destination node | Frame | NIC address | Bridges, Switches |
| **Physical** | Send signals, representing bits, across physical media | Bit | | Cables, Wireless Channels (Radio, Infrared, and Microwave), Connectors, Repeaters, and Hubs |

**Table 13.1**

Now we will discuss each device in detail.

## 13.3 Repeaters

Repeaters, working at the Physical Layer, are the simplest type of internetworking device. They are used to amplify electrical signals carried by the network. The function of a repeater is to receive incoming signals (a packet of data), regenerate the signals to their original strength, and retransmit them. Repeaters are used to lengthen individual network segments to form a larger extended network. That is, repeaters allow a network to be constructed that exceeds the size limit of a single physical segment by allowing additional lengths of cable to be connected (see figure 13.2).

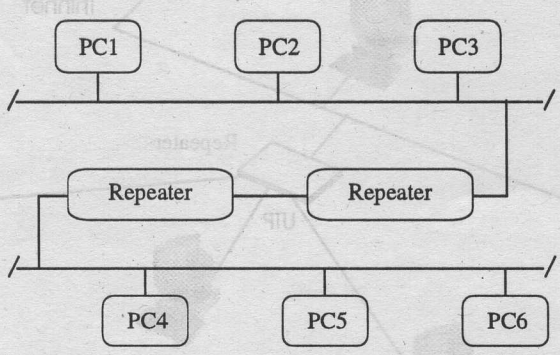



**Fig. 13.2** Repeaters used to lengthen individual network segments

### 13.3.1 How Repeaters Work ?

As nodes are added to a LAN, the cable can be extended with electrically passive connectors that simply attach one segment to another. Eventually, the permissible length of the cable segment will be exceeded, or it will become necessary to connect different cable types. A repeater is used to address these needs.

Because a repeater operates at the Physical Layer of the OSI model, the job of a repeater is to repeat bits. If a "1" bit is received on the input port of a repeater, a "1" bit is regenerated, with a stronger signal, at the output of the repeater. If a "0" bit is received on the input port of a repeater, a "0" bit is regenerated at the output of the repeater. A repeater is considered a non-discriminating device, because all incoming signals are passed on to each connected segment. Because a repeater reproduces exactly what it receives, bit-by-bit, it can reproduce errors. However, repeaters are very fast (10 megabits per second [Mbps] for Ethernet) and cause very little delay.

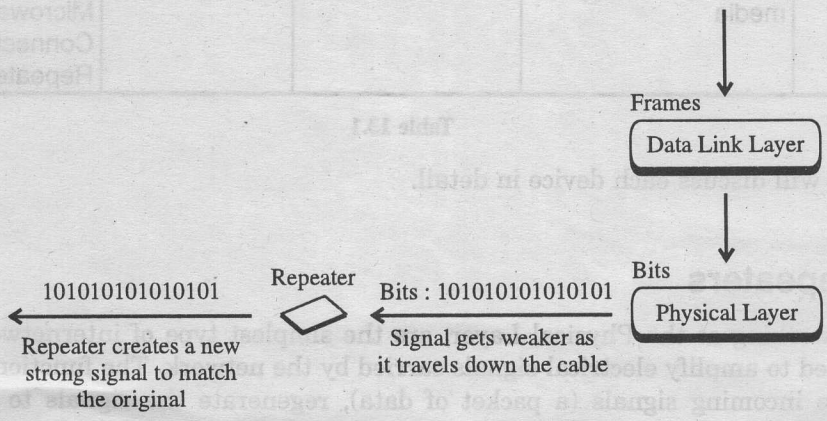



**Fig. 13.3** Repeater and OSI model

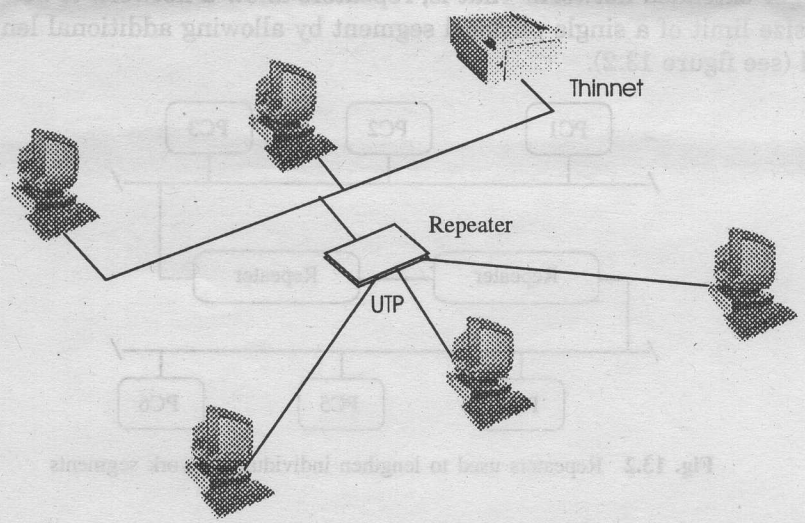



**Fig. 13.4** Repeater and media types

A repeater can connect one segment of a LAN to another, possibly connecting different types of media. For example, a repeater can connect thin Ethernet cables to UTP Ethernet cables, as shown in figure 13.4.

Because it is only a signal-boosting device, a repeater cannot connect two different media access types (Data Link protocols) such as Token Ring and Ethernet. For a repeater to be used both network segments must be same network protocols for all layers, same media access control method, and the same physical transmission technique. This means we could connect two segments that use the CSMA/CD access methods, or connect two segments that are running under the token-passing access method. However, we cannot connect a CSMA/CD segment to a token-passing segment.

Keep in mind that a repeater is a Physical Layer device. It cannot recognize the contents or format of a frame, and it cannot convert one type of Data Link header to another.

As internetworking devices for Ethernet LANs, repeaters are feasible only for relatively small LANs (less than 100 nodes), confined to a small geographical area such as one or two floors of an office building. A repeater should not be used to connect heavily used LANs because it cannot isolate traffic between LAN segments. Because each bit is copied to the attached segments, all data passes through a repeater in both directions. Therefore, if we connect multiple LAN segments using a repeater, we may experience performance problems, because total network traffic will increase.

## 13.4 Hubs

Hubs are argued as not being real internetworking devices. The reason we are including them in the internetworking portion is that hubs are often incorporated into other internetworking devices in order to minimize the amount of equipment installed at customer sites.

A hub is a box with a number of connectors to which multiple nodes (PCs) are attached. It serves as a common termination point that can relay signals along the appropriate paths. All hubs provide connectivity, and some even provide management capabilities. A hub usually connects nodes that have a common architecture.

Like repeaters, hubs operate at the Physical Layer, and regenerate signals bit-by-bit as they travel down the physical medium. Both repeaters and hubs provide signals greater reach;

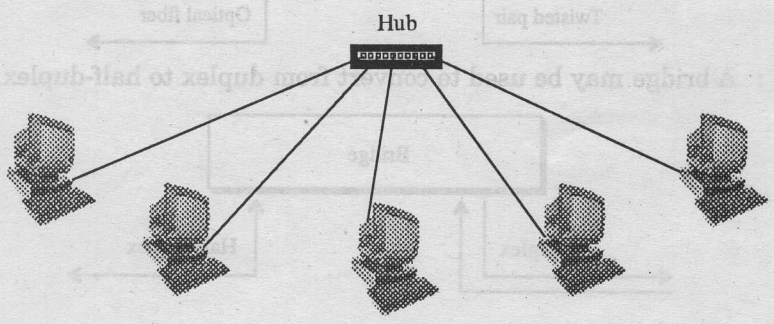



Fig. 13.5 Hub in a star topology

however, a hub also shares the same signals among a group of nodes. This clustering helps control the morass of cable and cable types found in many networks.

The Hub in figure 13.5 illustrates a simple hub configuration. It is important to realize that this structured wiring distinguishes between a network's physical and logical topologies. All hubs use a physical star topology; however, the logical topology of the network segment can be a ring, bus, tree or other structure, depending on the way a signal is passed from one node to the others.

## 13.5 Bridges

Bridges handle the first two layers of the OSI model-the physical layer and the data link layer. A hub is used to connect transmission medium of the same type and when the same MAC protocol is being used. To connect mediums of different types or to change between MAC protocols requires a bridge as shown in figure 13.6 and examples that follow.

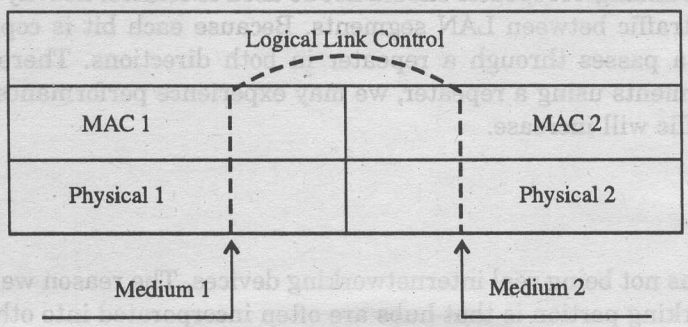
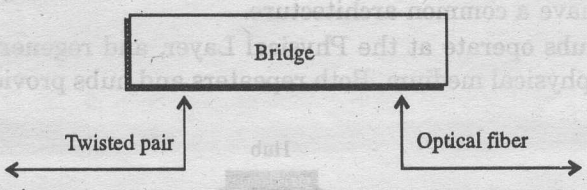



**Fig. 13.6** Bridge used to connect mediums of different types

***Example*** : A bridge may be used to convert from a copper wire (twisted pair) to an optical fiber medium.




***Example*** : A bridge may be used to convert from duplex to half-duplex.